RM0442
Reference manual

STM32MP153
advanced Arm®-based 32-bit MPUs

Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32MP153 microprocessor memory and peripherals.

The STM32MP153 is a family of microprocessors with different packages and peripherals.

Refer to the corresponding datasheets for ordering information, mechanical and electrical device characteristics.

For information on the Arm® Cortex®-A7 and Cortex®-M4 cores, refer to the Cortex®-A7 and Cortex®-M4 *Technical Reference Manuals*.

The STM32MP153 microprocessors include ST state-of-the-art patented technology.

Related documents

- Cortex®-A7 Technical Reference Manual, available from: http://infocenter.arm.com
- Cortex®-M4 Technical Reference Manual, available from: http://infocenter.arm.com
- STM32MP153x datasheet
- STM32 Cortex®-M4 MCUs and MPUs programming manual (PM0214)
- STM32MP151x/3x/7x device errata (ES0438)

Contents

List of tables

List of figures

1 Documentation conventions

1.1 General information

The STM32MP153 devices embed an Arm®[a] Cortex®-M4 with FPU and a dual Arm® Cortex®-A7 with FPU cores.

1.2 List of abbreviations for registers

The following abbreviations[b] are used in register descriptions:

read/write (rw)	Software can read and write to this bit.
read-only (r)	Software can only read this bit.
write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing 1 has no effect on the bit value.
read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing 0 has no effect on the bit value.
read/clear write (rc_w)	Software can read as well as clear this bit by writing to the register. The value written to this bit is not important.
read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to 0. Writing this bit has no effect on the bit value.
read/set by read (rs_r)	Software can read this bit. Reading this bit automatically sets it to 1. Writing this bit has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit. Writing 0 has no effect on the bit value.
read/write once (rwo)	Software can only write once to this bit and can also read it at any time. Only a reset can return the bit to its reset value.
toggle (t)	The software can toggle this bit by writing 1. Writing 0 has no effect.
read-only write trigger (rt_w1)	Software can read this bit. Writing 1 triggers an event but has no effect on the bit value.
Reserved (Res.)	Reserved bit, must be kept at reset value.

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

b. This is an exhaustive list of all abbreviations applicable to STMicroelectronics microcontrollers, some of them may not be used in the current document.

1.3 Register reset value

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is undefined are marked as X.

Bits (binary notation) or bits nibbles (hexadecimal notation) of which the reset value is unmodified are marked as U.

1.4 Glossary

This section gives a brief definition of acronyms and abbreviations used in this document:

- The product integrates two debug ports:
  - JTAG debug port (**JTAG-DP**) provides a 5-pin standard interface based on the Joint Test Action Group (JTAG) protocol.
  - SWD debug port (**SWD-DP**) provides a 2-pin (clock and data) interface based on the Serial Wire Debug (SWD) protocol.
- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Double word**: data of 64-bit length.
- **Option bytes**: product configuration bits stored in internal fuses.
- **OTP**: one-time programming (fuses).
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.
- **AXI**: Advanced extensible interface protocol.
- **ECC**: error code correction.
- **DMA**: direct memory access.
- **AHBS**: AHB slave bus.
- **AXIM**: AXI interconnect matrix.

1.5 Availability of peripherals

For availability of peripherals and their number across all devices, refer to the particular device datasheet.

2 Memory and bus architecture

2.1 System architecture

2.1.1 Bus architecture

The bus architecture is organized around two interconnect matrix, which operate in different frequency domains, as described in *Figure 1*.

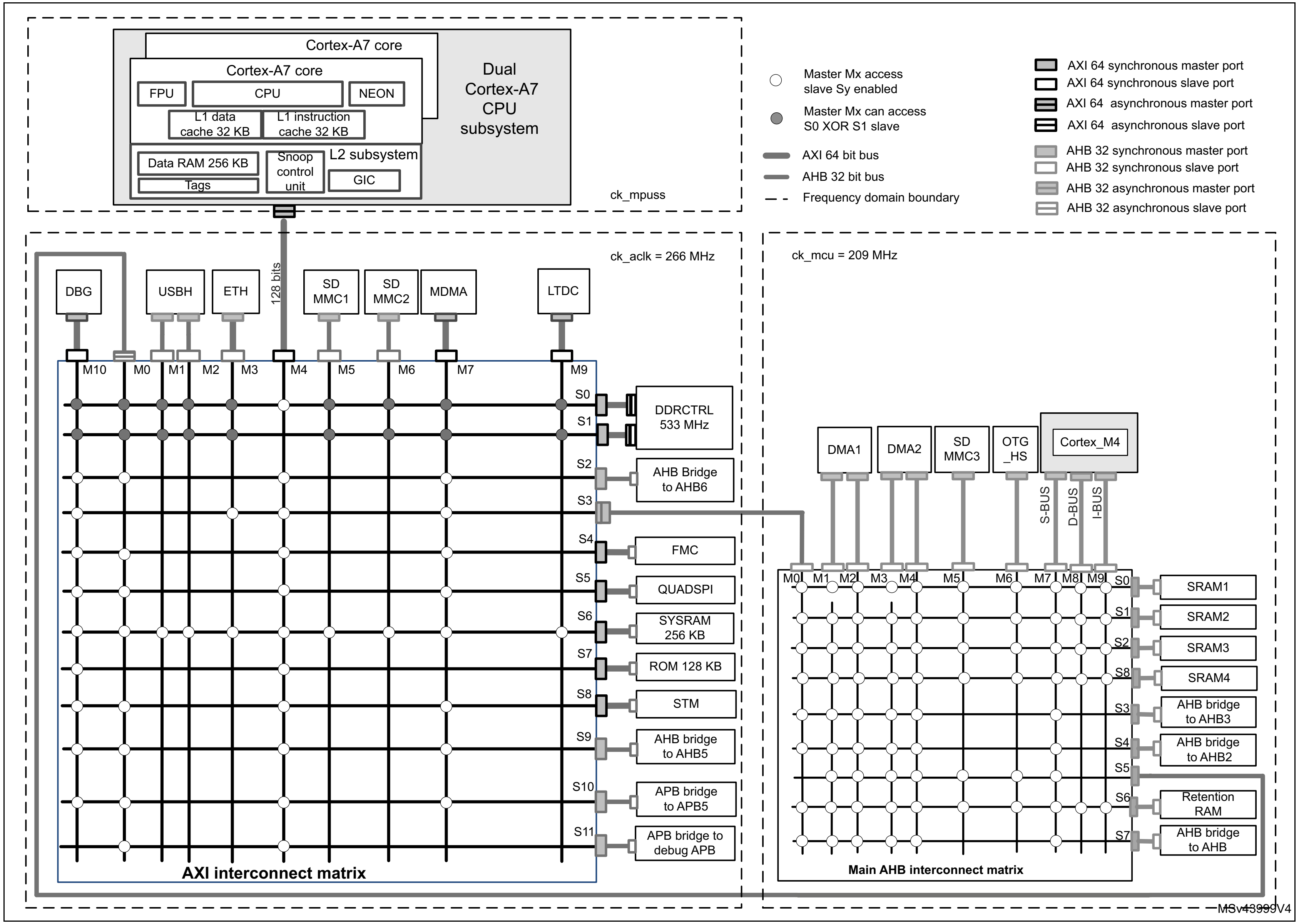
- One high speed **Arm® CoreLink™ NIC-400 network interconnect AXI-based**:
  This interconnect is dedicated to Cortex-A7 CPU subsystem and high bandwidth masters (this part of the system is called MPU-side in the following): USBH, ETH, SDMMC1/2, MDMA, LTDC.
  It manages accesses to:
  - external memories DDR (through DDRCTRL), NAND/NOR (through FMC or QUADSPI)
  - main system internal memories SYSRAM and ROM
  - main cluster of security peripherals through AHB5 bus
  - some low-speed communication peripherals through APB5 bus
  - control and configuration of the high-bandwidth masters
  - STM (system trace macrocell)
- One **multi-layer AHB interconnect** with an architecture inherited from former MCU families (this part of the system is called MCU-side in the following).
  This interconnect is dedicated to Cortex-M4 subsystem and associated masters: OTG, DMA1/2, SDMMC3.
  It manages accesses to:
  - Cortex-M4 dedicated SRAM1,2,3,4
  - most timers and low-speed communication peripherals through AHB2 bus
  - Cortex-M4 dedicated retention memory (RETRAM)
  - secondary cluster of security peripherals and camera interface through AHB3 bus
  - resources dedicated to communication with Cortex-A7 cluster: mailbox and semaphore through AHB3 bus
  - main infrastructure blocks as clock generation, power control, system configuration, and miscellaneous peripherals through AHB4 bus

AXI and multi-layer AHB matrix are connected to each other through one dedicated data-path in both directions:

- AXIM master port has access to all multi-layer AHB slaves. Some of them can become not accessible when the M4 isolation feature is activated.
- Multi-layer AHB master has access to all AXI slaves except:
  - ROM
  - slaves that can be accessed in secure mode only.

The bus architecture diagram in *Figure 1* describes the interconnect for STM32MP153 family.

Figure 1. Bus interconnect for STM32MP153



2.1.2 Memory Map organization

Introduction

For the detailed mapping of available memory and register areas, please refer to *Section 2.5: Memory organization*.

The following paragraphs describe the various resources in the system.

Embedded Memories

- **ROM**
  The 128 Kbyte embedded ROM is a fast access memory (no wait states). It is dedicated to Cortex-A7 CPU subsystem for boot code execution.This memory and the associated memory controller operate at aclk = 266 MHz.
- **Cortex-A7 CPU subsystem cache memories**
  Please refer to Cortex-A7 CPU subsystem description for cache memories characteristics.
- **SYSRAM**
  The 256 Kbyte embedded SYSRAM is a fast access static RAM, dedicated to Cortex-A7 CPU subsystem for code execution and low-latency data management. The SRAM itself is 0 wait-states. Some latency is added when data is propagated through the AXI interconnect.This memory and the associated memory controller operate at aclk = 266 MHz.
- **MCU SRAM1,2,3,4**
  The SRAM1,2,3,4 are two 128 Kbyte and two 64 Kbyte fast access static RAMs dedicated to Cortex-M4 MCU for code execution, low-latency data management, and data storage for exchange with the Cortex-A7 CPU subsystem.The SRAM1,2,3,4 are 0 wait-states. Some latency is added when data is propagated through the AHB interconnect. These memories and the associated memory controllers operate at mcu_ck = 209 MHz.
- **Retention RAM (RETRAM)**
  The RETRAM is a 64 Kbyte fast access static RAMs with specific supply providing data retention in all power modes including V_{BAT} and STANDBY. The RETRAM is 0 wait-states. Some latency is added when data is propagated through the AHB interconnect (access from MCU-side), or AXI and AHB interconnects (access from MPU-side).This memory and the associated memory controllers operate at mcu_ck = 209 MHz.
- **Backup RAM**
  The backup RAM is a 4 Kbyte static RAMs with dual-rail supply providing data backup for the whole system, whatever the power mode, with standard latency.This memory and the associated memory control operate at hclk5 = 266 MHz.
- **OTP**
  OTP is a secure one-time-programmed non volatile memory with 3072 effective bits in total. It contains system configuration information.This memory and the associated memory controllers operate at pclk5 = 133 MHz.

  OTP is auto-loaded after reset. Bits are held in shadow registers accessible at 133 MHz.

External memories support

- **DDR3/LPDDR2 controller (DDRCTRL)**

  The DDR supported types are LPDDR2, LPDDR3, DDR3/3L. Regarding external memory, 16 or 32 bits data width are possible, while internally, the DDRCTRL is connected to 64-bit data width interconnect. The maximum supported size is 1 Gbytes.

  The DDRCTRL is accessible through two AXI data ports, which have same characteristics. A system configuration register allows a given master to select which port to use, to balance traffic between the two ports.

  More information on memory characteristics and controller parametrization are available in *Section 5: DDR3/LPDDR2/LPDDR3 controller (DDRCTRL)* and *Section 7: DDR physical interface control (DDRPHYC)*.

  The DDRCTRL operates at aclk = 266 MHz. The external memory operates at up to pll2_r = 533 MHz depending on the system configuration.

- **Flexible memory controller (FMC) / QUADSPI**

  There is no internal Flash memory in the SoC. The system initial-boot-phase and self-initialization is supported by boot ROM and BSEC.

  External Flash memory is accessible through NAND configurable FMC, or through QUADSPI.

  The FMC also allows parallel connection of a SRAM/PSRAM instead of Flash memory if needed.

- **SD/SDIO/MMC card host interfaces (SDMMCx)**

  There is no internal Flash memory in the SoC. The system initial-boot-phase and self-initialization is supported by boot ROM and BSEC.

  External Flash memory such as SD card or *e*•MMC are accessible through SDMMCx

  SD, SDIO, eMMC devices are interfaced through three SDMMC controllers. Two of them are hooked to the AXIM interconnect and the third one to multi-layer AHB subsystem.

  These controllers are masters of the interconnect and originate the transfers to/from SD/SDIO/MMC memories.

Peripherals clusters

There are seven peripherals clusters in the system, briefly described hereafter.

Note: the AHBx/APBy notation means that the resources are accessed through bus AHBx, and for some of them through bus AHBx in combination with bus APBy.

- **AHB4/APB3**
  This cluster groups important system functions as clock and power control, system configuration, most GPIOs control, plus some additional timers and low-speed communication interfaces.
- **AHB3**
  This cluster groups some security blocks dedicated to Cortex-M4, camera interface and interprocessor communication resources.
- **AHB2/APB1-APB2**
  This cluster groups most of the usual MCU peripherals: timers, low-speed communication interfaces (SPI, I2C, U(S)ART, SAI), ADC, DAC, FDCAN and DFSDM.
  All those IPs are typically controlled by Cortex-M4, directly or through DMA1/2, but can also be managed by Cortex-A7 CPU subsystem as well.
- **AHB5**
  This cluster groups security blocks dedicated to Cortex-A7 and the backup RAM.
- **APB5**
  This cluster groups some low-speed secure communication blocks dedicated for Cortex-A7, in addition to BSEC and chip configuration control resources.
  Real time clock generation and anti-tamper features also belong to this cluster.
- **AHB6/APB4**
  This cluster groups most of the graphic/display resources control, as well as access to configuration registers of the fast masters: MDMA, ETHernet, SDMMC1/2,USBH.
- **Debug**
  This cluster groups all debug features including Arm® CoreSight™ infrastructure.

Memory map overview

Table 1 summarizes which master (cf *Figure 1*) can access which memory-mapped resource.

Table 1. Memory map overview STM32MP153

Resource	Cortex-A7	Cortex-M4	De-bugger	MDMA / DMA1 / DMA2	ETH	OTG / SDMMC3	USBH / SDMMC1,2 / LTDC
DDR	x	x	x	x	x	x	x
Cortex-A7 debug	x(1)	-	x	-	-	-	-
Cortex-M4 debug	-	x	x	-	-	-	-
Cortex-A7 GIC	x	-	-	-	-	-	-
STM	x	x	x	x	-	-	-
Ext FMC interface	x	x	x	x	-	-	-
Ext QUADSPI	x	x	x	x	-	-	-
Ext NOR interface	x	x	x	x	-	-	-
AXIM registers	x	-	x	-	-	-	-
APB5 peripherals	x	x	x	x	-	-	-

Table 1. Memory map overview STM32MP153 (continued)

Resource	Cortex-A7	Cortex-M4	De-bugger	MDMA / DMA1 / DMA2	ETH	OTG / SDMMC3	USBH / SDMMC1,2 / LTDC
AHB6/APB4 peripherals	x	x	x	x	-	-	-
AHB5 peripherals	x	x	x	x	-	-	-
APB debug peripherals	x	x	x	-	-	-	-
AHB4/APB3 peripherals	x	x	x	x	x	-	-
AHB2/APB1,2 peripherals	x	x	x	x	x	-	-
AHB3/APB1,2 peripherals	x	x	x	x	x	-	-
RETRAM	x	x	x	x	x	x	-
SRAM 1,2,3,4	x	x	x	x	x	x	-
SYSRAM	x	x	x	x	x	x	x
ROM/RETRAM @0x0000 0000	x(2)	x(2)	x(2)	-	-	-	-

1. Cortex-A7 accesses its own debug resources through CoreSight infrastructure.
2. Cortex-A7 and debugger see the ROM, while the Cortex-M4 access RETRAM at this location.

2.2 AXI interconnect matrix (AXIM)

2.2.1 AXIM features

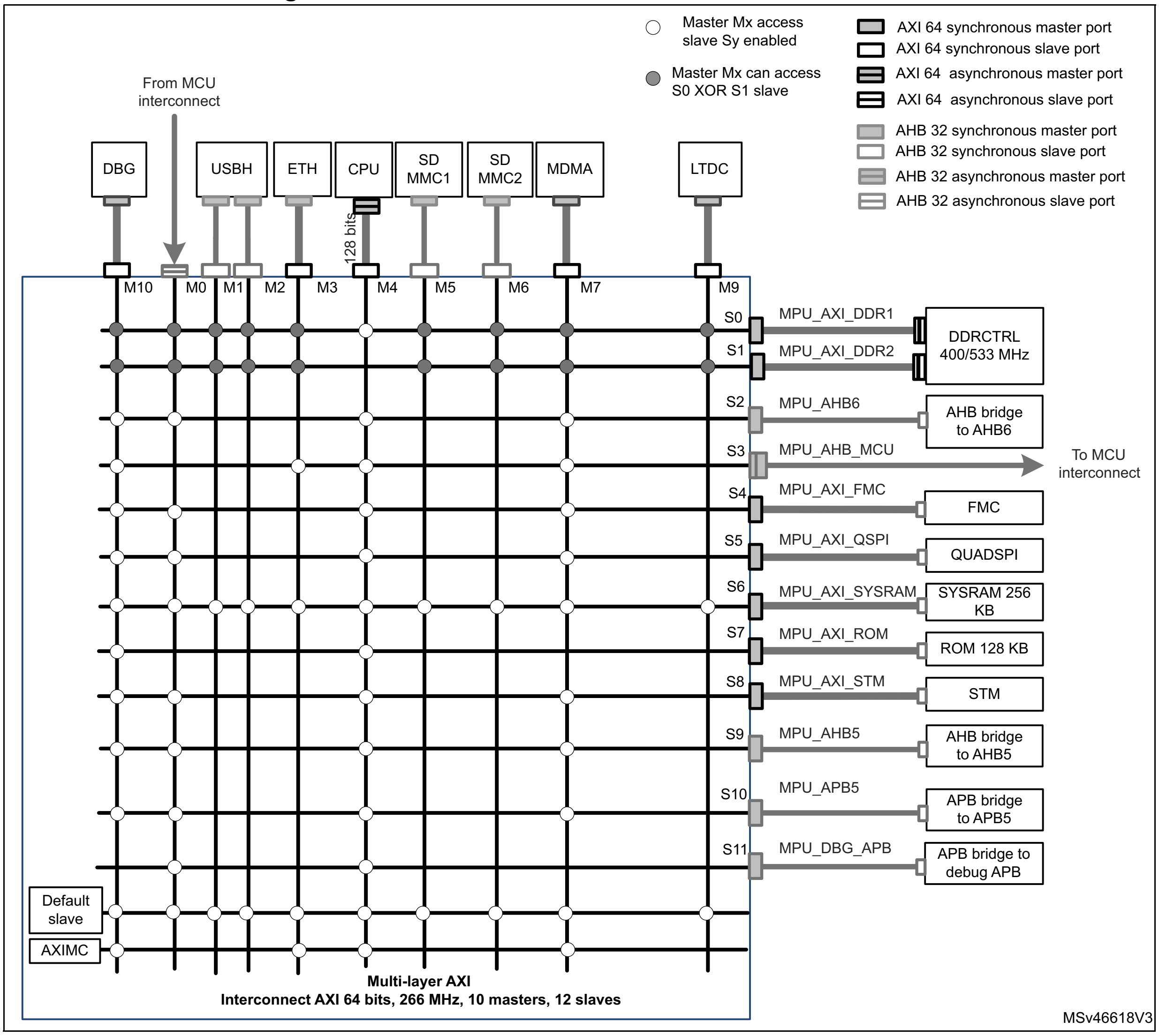
The AXIM is implemented with NIC-400 network interconnect. Its main features are:

- 10 masters, and 12 slave layers (plus 2 internal slaves), with optimized interconnect topology from masters to slaves (no full interconnect)
- Simultaneous accesses support from different masters to slaves connected to separated layers
- Protocol conversion (AXI, AHB) support
- Internal data and commands buffering for improved traffic efficiency
- QoS programming support
- Asynchronism management
- Software configuration of internal parameters (QOS, bypass,...) through user accessible registers.

2.2.2 AXIM interconnect configuration

The NIC-400 interconnect configuration is described in *Figure 2*.

Figure 2. NIC-400 interconnect AXIM for STM32MP153



2.2.3 Master ports description

- **M0**: asynchronous port connected to MCU interconnect master port
  Allows masters from the MCU subsystem to have access to all blocks in the system, except ROM and AXIMC internal slave (also known as GPV, global programmer view) which contains all NIC-400 programming registers.
- **M1/M2**: two AHB-Lite ports connected to USBHost master ports, with same capacities
  Accesses are limited to DDR and SYSRAM memories.
- **M3**: AXI4 port connected to ETHernet gigabit interface
  Accesses are limited to DDR and SYSRAM memories, plus memories located in MCU subsystem.
- **M4**: AXI3 port connected to Cortex-A7 CPU subsystem (128-bit wide)
  The asynchronism is managed inside the CPU subsystem. The conversion to 64 bits is done inside the NIC-400. The CPU subsystem is connected to DDR controller port 0 only. The user can choose to hook other masters on this layer, or leave the CPU as single master, depending on application needs.
- **M5/M6**: two AHB-Lite ports connected to SDMMC1 and SDMMC2 master ports
  Accesses are limited to DDR, and SYSRAM memories.
- **M7**: AXI4 port connected to MDMA
  MDMA is able to access all slaves in the system including the configuration ports of master IPs. It can be used to perform the entire programing of such blocks thanks to its linked list support. This master has important read issuing capabilities and is able to generate peak traffic approaching the layer theoretical max throughput.
- **M9**: AXI4 port connected to LTDC display driver
  Accesses are limited to DDR and SYSRAM memories.
- **M10**: AXI4 port connected to debug access port (DAP) AXI
  This master port has access to all slaves, except S11 going back to the DAP to avoid dual data-path in the system topology. M10 also accesses the AXIMC internal slave which contains all interconnect programming registers.

2.2.4 Master ports main characteristics

Table 2 summarizes the master ports main characteristics.

Table 2. Master ports main characteristics

Master port	Nb	Protocol	Data bus width	Synchronicity	ID width	Default QOS value(1)	Read/write issuing capability
MCU interco	M0	AHB_Lite	32	Async	1	6	1/1
USBH p1	M1	AHB_Lite	32	Sync	1	5	1/1
USBH p2	M2	AHB_Lite	32	Sync	1	5	1/1
ETH	M3	AXI4	64	Sync	4	7	1/1
CPU	M4	AXI3	128	Sync	6	12	8/8
SDMMC1	M5	AHB_Lite	32	Sync	1	4	1/1
SDMMC2	M6	AHB_Lite	32	Sync	1	4	1/1
MDMA	M7	AXI4	64	Sync	1	8	8/1
LTDC	M9	AXI4	64	Sync	4	11	2/1
DEBUG / DAP	M10	AXI4	64	Sync	1	2	1/1

1. Master priorities can be modified by software.

2.2.5 Master ports security characteristics

Table 3 lists the master ports security characteristics.

Table 3. Master ports security characteristics

Master ports	Nb	NIC input signals used for security propagation	TrustZone configuration	NSAID(1)
MCU interco	M0	HAuser(5:0)	Non-secure	0b0001 for Cortex-M4 ob0110 for DMA1 / DMA2 0b1000 for OTG
USBH p1	M1	HAuser(5:0)	Non-secure	0b0111
USBH p2	M2	HAuser(5:0)	Non-secure	0b0111
ETH	M3	ARuser(5:0) AWuser(5:0)	Non-secure	0b1010
CPU	M4	ARuser(5:0) AWuser(5:0)	Per-access	0b0000
SDMMC1	M5	HAuser(5:0)	Non-secure	0b1001
SDMMC2	M6	HAuser(5:0)	Non-secure	0b1001
MDMA	M7	ARuser(5:0) AWuser(5:0)	Per-access	0b0101
LTDC	M9	ARuser(5:0) AWuser(5:0)	Non-secure	0b0011
DEBUG / DAP	M10	ARuser(5:0) AWuser(5:0)	Per-access	0b1111

1. Master M0 transmits several NSAIDs corresponding to multi-layer AHB masters. Different masters of same type (SDMMC1/2 and USBH1/2) generate the same NSAID value.

2.2.6 Slave ports description

- **S0, S1**: AXI3 synchronous ports 0 and 1 of DDR controller. Asynchronism is managed inside the DDR controller.
- **S2**: AHB-Lite synchronous port connected to AHB6, which is the bridge to configuration ports of all masters.
- **S3**: AHB-Lite asynchronous port connected to main MCU multi-layer AHB interconnect, giving access to Cortex-A7 to all MCU peripherals.
- **S4**: AXI3 synchronous port to NAND/NOR controller
- **S5**:AXI3 synchronous port to QUADSPI controller
- **S6**:AXI3 synchronous port to SYSRAM controller
- **S7**:AXI3 synchronous port to ROM controller
- **S8**: AXI3 synchronous port to STM trace data buffer
- **S9**: AHB-Lite synchronous port to AHB5. This bridge controls access to crypto resources, secure GPI0s and backup RAM.
- **S10**: AHB-Lite synchronous port to protocol converter for APB5. APB5 is an APB3 bus supporting security signals propagation. This bus controls access to security management blocks and a set of secure peripherals.
- **S11**: AHB-Lite synchronous port to protocol converter for APB_DEBUG access located in DAP.

Internal slaves:

- **Default slave**: defines the interconnect behavior when accesses are done in a non-valid region.
- **AXIMC** (also known as GPV): contains all NIC-400 interconnect configuration registers. The interconnect configuration register set can be accessed only by two masters: Cortex-A7 CPU (software configuration in early stage of application) and DAP (debug, mainly for check).

2.2.7 Slave ports main characteristics

Table 4 lists the slave ports main characteristics.

Table 4. Slave ports main characteristics

Slave port	Nb	Protocol	Data bus width	Synchronicity	Read / write acceptance capability
DDR p0	S0	AXI4	64	Sync	20/20(1)
DDR p1	S1	AXI4	64	Sync	20/20(1)
AHB6	S2	AHB_Lite	32	Sync	1/1
Multi-layer AHB	S3	AXI4	32	Async	1/1
FMC	S4	AXI4	64	Sync	1/1
QUADSPI	S5	AXI4	64	Sync	1/1
SYSRAM	S6	AXI3	64	Sync	1/1
ROM	S7	AXI3	64	Sync	1/1
STM	S8	AXI4	64	Sync	1/1
AHB5	S9	AXI4	32	Sync	1/1
APB5	S10	AHB_Lite	32	Sync	1/1
DEBUG/DAP	S11	AHB_Lite	32	Sync	1/1

1. DDR read / write acceptance capability = CAM size (16) + internal AXI port interface FIFO (4)

2.2.8 Slave ports security characteristics

Table 5 lists the slave ports security characteristics.

Table 5. Slave ports security characteristics

Slave ports	Nb	NIC input signals used for security propagation	TrustZone configuration
DDR p0	S0	ARuser(5:0) AWuser(5:0)	Secure and non-secure
DDR p1	S1	ARuser(5:0) AWuser(5:0)	Secure and non-secure
AHB6	S2	HAuser(5:0)	Secure and non-secure
Multi-layer AHB	S3	HAuser(5:0)	Secure and non-secure
FMC	S4	ARuser(5:0) AWuser(5:0)	Secure and non-secure
QUADSPI	S5	HAuser(5:0)	Secure and non-secure
SYSRAM	S6	ARuser(5:0) AWuser(5:0)	Secure and non-secure
ROM	S7	ARuser(5:0) AWuser(5:0)	Secure and non-secure
STM	S8	ARuser(5:0) AWuser(5:0)	Secure and non-secure
AHB5	S9	ARuser(5:0) AWuser(5:0)	Secure and non-secure
APB5	S10	ARuser(5:0) AWuser(5:0)	Secure and non-secure
DEBUG/DAP	S11	HAuser(5:0)	Per-access

2.3 Multi-layer AHB interconnect

2.3.1 Multi-layer AHB features

The multi-layer AHB interconnect is an optimized version of Arm BP010 interconnect.

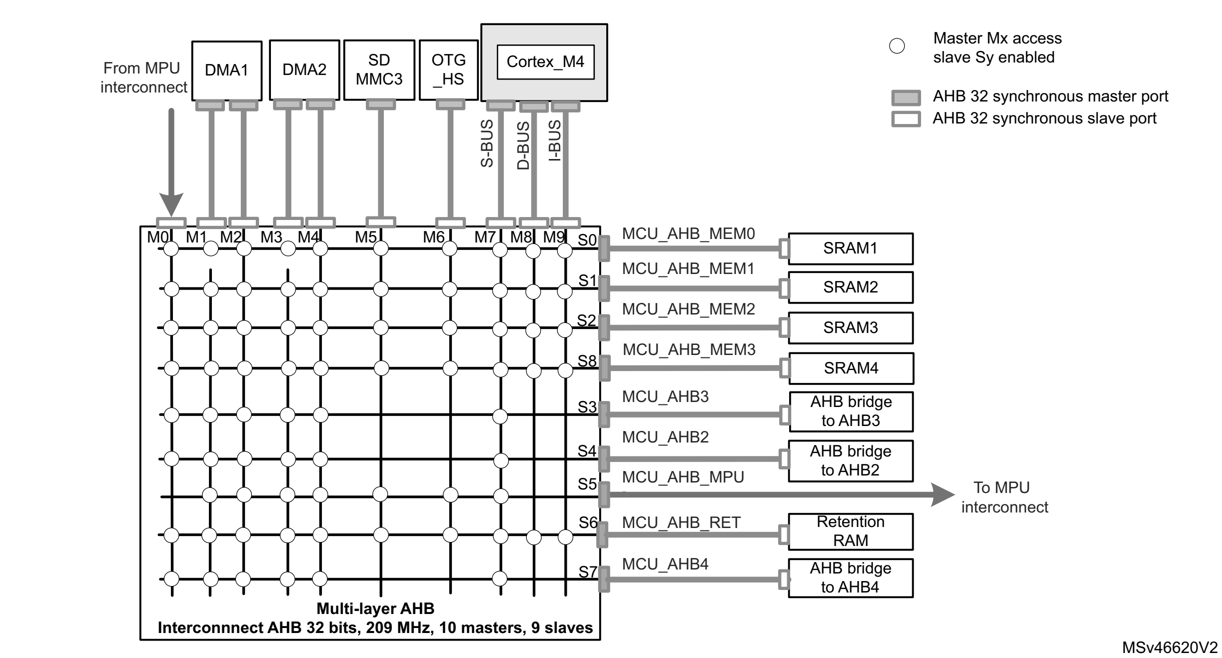
Its main features are:

- 10 masters, and 9 slave layers, all 32-bit data
- Simultaneous accesses support from different masters to slaves connected to separated layers
- Internal data and commands buffering for improved traffic efficiency

2.3.2 Multi-layer AHB interconnect configuration

The multi-layer AHB interconnect configuration is described in *Figure 3*.

Figure 3. Multi-layer AHB interconnect for STM32MP153



2.3.3 Multi-layer AHB master ports characteristics

- **M0**: AHB-Lite connected to MPU interconnect master port
  It allows masters from the MPU subsystem to have access to all slaves in the system, except the path to MPU to avoid loopback.
- **M1, M2, M3, M4**: four AHB-Lite ports connected to DMA1 and DMA2 dual-master blocks, with same capacities.
  Accesses are granted to all slaves in the subsystem: Cortex-M4 SRAMs and bridges to all peripherals.
- **M5**: AHB-Lite ports connected to SDMMC3 interface.
  Accesses are limited to Cortex-M4 RAMs, retention RAM and memories located in MPU subsystem (SYSRAM, DDR) through the main NIC-400 AXI interconnect.
- **M6**: AHB-Lite port connected to OTG interface.
  Accesses are limited to Cortex-M4 RAMs, retention RAM and memories located in MPU subsystem (SYSRAM, DDR) through the main NIC-400 AXI interconnect.
- **M7**: AHB-Lite port connected to the S-bus Cortex-M4 subsystem.
  It allows the Cortex-M4 to have access to all slaves in the system, including a number of those located in MPU subsystem through the main NIC-400 AXI interconnect.
- **M8, M9**: AHB-Lite port connected to the I-bus and D-bus of Cortex-M4 subsystem.
  It allows the Cortex-M4 to have access to slaves in the system located below 0x20 000 000, which are retention RAM and SRAM1,2,3,4. No peripherals are re-mapped below 0x20 000 000.

Table 6 lists the main characteristics of multi-layer AHB master ports.

Table 6. Multi-layer AHB master ports main characteristics

Master ports	Nb	Protocol	Data bus width	Synchronicity
AHB_MPU	M0	AHB-Lite	32	Sync
DMA1 p1	M1	AHB-Lite	32	Sync
DMA1 p2	M2	AHB-Lite	32	Sync
DMA2 p1	M3	AHB-Lite	32	Sync
DMA2 p2	M4	AHB-Lite	32	Sync
SDMMC3	M5	AHB-Lite	32	Sync
OTG	M6	AHB-Lite	32	Sync
SBUS of Cortex-M4	M7	AHB-Lite	32	Sync
DBUS of Cortex-M4	M8	AHB-Lite	32	Sync
IBUS of Cortex-M4	M9	AHB-Lite	32	Sync

2.3.4 Multi-layer AHB slave ports characteristics

- **S0, S1, S2, S8**: AHB-Lite synchronous ports to SRAM1,2,3,4.
  S1 is the slave which provides the best latency when accessed from M4 S-port.
- **S3**: AHB-Lite synchronous port to AHB3
  This bridge controls access to crypto resources and communications IPs with Cortex-A7 cluster.
- **S4**: AHB-Lite synchronous port connected to AHB2
  It is the bridge to configuration ports of all masters connected to multi-layer AHB. This port also allows to access all resources located in APB1 and APB2 clusters, and ADCs.
- **S5**: AHB-Lite synchronous port connected to main NIC-400 interconnect
  It gives access to multi-layer AHB master to most SoC peripherals.
- **S6**: AHB-Lite synchronous port to retention RAM controller
- **S7**: AHB-Lite synchronous port to AHB4
  This bridge controls access to a cluster comprising main system management utilities (clock, power), all non-secure GPIOs and a complementary set of peripherals, allowing to activate only this cluster for simple applications.

The following table lists the multi-layer AHB slave ports main characteristics.

Table 7. Multi-layer AHB slave ports main characteristics

Slave port	Nb	Protocol	Data bus width	Synchronicity	Clock ratio
SRAM1	S0	AHB-Lite	32	sync	1/1
SRAM2	S1	AHB-Lite	32	sync	1/1
SRAM3	S2	AHB-Lite	32	sync	1/1
SRAM4	S8	AHB-Lite	32	sync	1/1
AHB3	S3	AHB-Lite	32	sync	1/1
AHB2	S4	AHB-Lite	32	sync	1/1
MPU_NIC400	S5	AHB-Lite	32	sync	1/1
RETRAM	S6	AHB-Lite	32	sync	1/1
AHB4	S7	AHB-Lite	32	sync	1/1

2.4 AXIMC registers

The following register description is only about AXIM interconnect, as multi-layer AHB interconnect has no configuration registers.

x is the number of the master (Mx with x=10 means M10). Parameters of the masters vary depending on their capabilities, so the number of registers may differ from one to another.

2.4.1 AXIMC peripheral ID4 register (AXIMC_PERIPH_ID_4)

Address offset: 0x1FD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	K4COUNT[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **K4COUNT[3:0]:** register file size

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code

2.4.2 AXIMC peripheral ID5 register (AXIMC_PERIPH_ID_5)

Address offset: 0x1FD4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_5[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PERIPH_ID_5[7:0]:** reserved, not used.

2.4.3 AXIMC peripheral ID6 register (AXIMC_PERIPH_ID_6)

Address offset: 0x1FD8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_6[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PERIPH_ID_6[7:0]:** reserved, not used.

2.4.4 AXIMC peripheral ID7 register (AXIMC_PERIPH_ID_7)

Address offset: 0x1FDC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_7[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PERIPH_ID_7[7:0]:** reserved, not used.

2.4.5 AXIMC peripheral ID0 register (AXIMC_PERIPH_ID_0)

Address offset: 0x1FE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PERIPH_ID_0[7:0]:** part number [7:0]

2.4.6 AXIMC peripheral ID1 register (AXIMC_PERIPH_ID_1)

Address offset: 0x1FE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_1[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PERIPH_ID_1[7:0]:** JEP106 identity [3:0], part number [11:8]

2.4.7 AXIMC peripheral ID2 register (AXIMC_PERIPH_ID_2)

Address offset: 0x1FE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PERIPH_ID_2[7:0]:** part revision, JEP106 code flag, JEP106 identity [6:4]

2.4.8 AXIMC peripheral ID3 register (AXIMC_PERIPH_ID_3)

Address offset: 0x1FEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_AND[3:0]				CUST_MOD_NUM[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REV_AND[3:0]:** customer version
0: none

Bits 3:0 **CUST_MOD_NUM[3:0]:** customer modification
0: none

2.4.9 AXIMC component ID0 register (AXIMC_COMP_ID_0)

Address offset: 0x1FF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]:** preamble bits [7:0]

2.4.10 AXIMC component ID1 register (AXIMC_COMP_ID_1)

Address offset: 0x1FF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Component class
0xF: generic IP component class

Bits 3:0 **PREAMBLE[11:8]:** preamble bits [11:8]
0x0: common ID value

2.4.11 AXIMC component ID2 register (AXIMC_COMP_ID_2)

Address offset: 0x1FF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]:** preamble bits [19:12]

2.4.12 AXIMC component ID3 register (AXIMC_COMP_ID_3)

Address offset: 0x1FFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]:** preamble bits [27:20]

2.4.13 AXIMC master x packing functionality register (AXIMC_Mx_FN_MOD2)

Address offset: 0x42024 + 0x1000 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS_MERGE:** Disable packing of beats to match the output data width
Unaligned transactions are not realigned to the input data word boundary.
0: normal operation
1: disable packing

2.4.14 AXIMC master x AHB conversion override functionality register (AXIMC_Mx_FN_MOD_AHB)

Address offset: 0x42028 + 0x1000 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WR_INC_OVERRIDE**: Converts all AHB-Lite write transactions to a series of single beat AXI transactions, and each AHB-Lite write beat is acknowledged with the AXI buffered write response.
0: override disabled
1: override enabled

Bit 0 **RD_INC_OVERRIDE**: Converts all AHB-Lite read transactions to a series of single beat AXI transactions.
0: override disabled
1: override enabled

2.4.15 AXIMC master x read priority register (AXIMC_Mx_READ_QOS)

Address offset: 0x42100 + 0x1000 * x, (x = 0 to 2)

Reset value: Block 0: 0x0000 0006

Reset value: Block 1: 0x0000 0005

Reset value: Block 2: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AR_QOS[3:0]:** read channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.16 AXIMC master x write priority register (AXIMC_Mx_WRITE_QOS)

Address offset: 0x42104 + 0x1000 * x, (x = 0 to 2)

Reset value: Block 0: 0x0000 0006

Reset value: Block 1: 0x0000 0005

Reset value: Block 2: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AW_QOS[3:0]:** write channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.17 AXIMC master x issuing capability override functionality register (AXIMC_Mx_FN_MOD)

Address offset: 0x42108 + 0x1000 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE:** override AMIB write issuing capability
0: normal issuing capability
1: force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE:** override AMIB read issuing capability
0: normal issuing capability
1: force issuing capability to 1

2.4.18 AXIMC master x packing functionality register (AXIMC_Mx_FN_MOD2)

Address offset: 0x42024 + 0x1000 * x, (x = 5 to 6)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS_MERGE:** Disable packing of beats to match the output data width
Unaligned transactions are not realigned to the input data word boundary.
0: normal operation
1: disable packing

2.4.19 AXIMC master x AHB conversion override functionality register (AXIMC_Mx_FN_MOD_AHB)

Address offset: 0x42028 +0x1000 * x, (x = 5 to 6)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WR_INC_OVERRIDE**: converts all AHB-Lite write transactions to a series of single beat AXI transactions, and each AHB-Lite write beat is acknowledged with the AXI buffered write response.
0: override disabled
1: override enabled

Bit 0 **RD_INC_OVERRIDE**: converts all AHB-Lite read transactions to a series of single beat AXI transactions.
0: override disabled
1: override enabled

2.4.20 AXIMC master x read priority register (AXIMC_Mx_READ_QOS)

Address offset: 0x42100 +0x1000 * x, (x = 5 to 6)

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AR_QOS[3:0]:** read channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.21 AXIMC master x write priority register (AXIMC_Mx_WRITE_QOS)

Address offset: 0x42104 + 0x1000 * x, (x = 5 to 6)

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AW_QOS[3:0]:** write channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.22 AXIMC master x issuing capability override functionality register (AXIMC_Mx_FN_MOD)

Address offset: 0x42108 + 0x1000 * x, (x = 5 to 6)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE:** override AMIB write issuing capability
0: normal issuing capability
1: force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE:** override AMIB read issuing capability
0: normal issuing capability
1: force issuing capability to 1

2.4.23 AXIMC master x read priority register (AXIMC_Mx_READ_QOS)

Address offset: 0x42100 + 0x1000 * x, (x = 3 to 4)

Reset value: Block 3: 0x0000 0007

Reset value: Block 4: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AR_QOS[3:0]:** read channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.24 AXIMC master x write priority register (AXIMC_Mx_WRITE_QOS)

Address offset: 0x42104 + 0x1000 * x, (x = 3 to 4)

Reset value: Block 3: 0x0000 0007

Reset value: Block 4: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AW_QOS[3:0]:** write channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.25 AXIMC master x packing functionality register (AXIMC_Mx_FN_MOD)

Address offset: 0x42108 + 0x1000 * x, (x = 3 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE:** override AMIB write issuing capability
0: normal issuing capability
1: force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE:** override AMIB read issuing capability
0: normal issuing capability
1: force issuing capability to 1

2.4.26 AXIMC master x read priority register (AXIMC_Mx_READ_QOS)

Address offset: 0x42100 + 0x1000 * x, (x = 7 to 8)

Reset value: Block 7: 0x0000 0008

Reset value: Block 8: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AR_QOS[3:0]:** read channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.27 AXIMC master x write priority register (AXIMC_Mx_WRITE_QOS)

Address offset: 0x42104 + 0x1000 * x, (x = 7 to 8)

Reset value: Block 7: 0x0000 0008

Reset value: Block 8: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AW_QOS[3:0]:** write channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.28 AXIMC master x issuing capability override functionality register (AXIMC_Mx_FN_MOD)

Address offset: 0x42108 + 0x1000 * x, (x = 7 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE:** override AMIB write issuing capability
0: normal issuing capability
1: force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE:** override AMIB read issuing capability
0: normal issuing capability
1: force issuing capability to 1

2.4.29 AXIMC long burst capability inhibition register (AXIMC_FN_MOD_LB)

Address offset: 0x4A02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **FN_MOD_LB**: controls burst breaking of long bursts
0: long bursts can not be generated at the output of the ASIB
1: long bursts can be generated at the output of the ASIB

2.4.30 AXIMC master x read priority register (AXIMC_Mx_READ_QOS)

Address offset: 0x42100 + 0x1000 * x, (x = 9 to 10)

Reset value: Block 9: 0x0000 000B

Reset value: Block 10: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AR_QOS[3:0]:** read channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.31 AXIMC master x write priority register (AXIMC_Mx_WRITE_QOS)

Address offset: 0x42104 + 0x1000 * x, (x = 9 to 10)

Reset value: Block 9: 0x0000 000B

Reset value: Block 9: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **AW_QOS[3:0]:** write channel QoS setting
0x0: lowest priority
0xF: highest priority

2.4.32 AXIMC master x issuing capability override functionality register (AXIMC_Mx_FN_MOD)

Address offset: 0x42108 + 0x1000 * x, (x = 9 to 10)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE_ISS_OVERRIDE	READ_ISS_OVERRIDE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITE_ISS_OVERRIDE:** override AMIB write issuing capability
0: normal issuing capability
1: force issuing capability to 1

Bit 0 **READ_ISS_OVERRIDE:** override AMIB read issuing capability
0: normal issuing capability
1: force issuing capability to 1

2.4.33 AXIMC register map

Table 8. AXIMC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1FD0	AXIMC_PERIPH_ID_4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	K4COUNT[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0x1FD4	AXIMC_PERIPH_ID_5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_5[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1FD8	AXIMC_PERIPH_ID_6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_6[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1FDC	AXIMC_PERIPH_ID_7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_7[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1FE0	AXIMC_PERIPH_ID_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x1FE4	AXIMC_PERIPH_ID_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_1[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x1FE8	AXIMC_PERIPH_ID_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIPH_ID_2[7:0]							
	Reset value																									0	0	1	1	1	0	1	1
0x1FEC	AXIMC_PERIPH_ID_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_AND[3:0]				CUST_MOD_NUM[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0x1FF0	AXIMC_COMP_ID_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0x1FF4	AXIMC_COMP_ID_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0x1FF8	AXIMC_COMP_ID_2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0x1FFC	AXIMC_COMP_ID_3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1
0x2000 - 0x42020	Reserved	Reserved																															

Table 8. AXIMC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x42024	AXIMC_M0_FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x42028	AXIMC_M0_FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4202C - 0x420FC	Reserved	Reserved																															
0x42100	AXIMC_M0_READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	1	1	0
0x42104	AXIMC_M0_WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	1	1	0
0x42108	AXIMC_M0_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4210C - 0x43020	Reserved	Reserved																															
0x43024	AXIMC_M1_FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x43028	AXIMC_M1_FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4302C - 0x430FC	Reserved	Reserved																															

Table 8. AXIMC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x43100	**AXIMC_M1_READ_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	1	0	1
0x43104	**AXIMC_M1_WRITE_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	1	0	1
0x43108	**AXIMC_M1_FN_MOD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4310C - 0x44020	Reserved	Reserved																															
0x44024	**AXIMC_M2_FN_MOD2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x44028	**AXIMC_M2_FN_MOD_AHB**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4402C - 0x440FC	Reserved	Reserved																															
0x44100	**AXIMC_M2_READ_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	1	0	1
0x44104	**AXIMC_M2_WRITE_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	1	0	1

Table 8. AXIMC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44108	AXIMC_M2_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4410C - 0x450FC	Reserved	Reserved																															
0x45100	AXIMC_M3_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	1	1	1
0x45104	AXIMC_M3_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	1	1	1
0x45108	AXIMC_M3_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4510C - 0x460FC	Reserved	Reserved																															
0x46100	AXIMC_M4_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													1	1	0	0
0x46104	AXIMC_M4_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													1	1	0	0
0x46108	AXIMC_M4_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4C10C - 0x47020	Reserved	Reserved																															

Table 8. AXIMC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x47024	AXIMC_M5_FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x47028	AXIMC_M5_FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4702C - 0x470FC	Reserved	Reserved																															
0x47100	AXIMC_ M5_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	1	0	0
0x47104	AXIMC_M5_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	1	0	0
0x47108	AXIMC_M5_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4710C - 0x48020	Reserved	Reserved																															
0x48024	AXIMC_M6_FN_MOD2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS_MERGE
	Reset value																																0
0x48028	AXIMC_M6_FN_MOD_AHB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_INC_OVERRIDE	RD_INC_OVERRIDE
	Reset value																															0	0
0x4802C - 0x480FC	Reserved	Reserved																															

Table 8. AXIMC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x48100	AXIMC_M6_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	1	0	0
0x48104	AXIMC_M6_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	1	0	0
0x48108	AXIMC_M6_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4810C - 0x490FC	Reserved	Reserved																															
0x49100	AXIMC_M7_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													1	0	0	0
0x49104	AXIMC_M7_ WRITE_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													1	0	0	0
0x49108	AXIMC_M7_FN_MOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD[1:0]	
	Reset value																															0	0
0x4910C - 0x4A028	Reserved	Reserved																															
0x4A02C	AXIMC_M8_FN_MOD_ LB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD_LB
	Reset value																																0
0x4A030 - 0x4A0FC	Reserved	Reserved																															
0x4A100	AXIMC_M8_ READ_QOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	0	1	1

Table 8. AXIMC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4A104	**AXIMC_M8_WRITE_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	0	1	1
0x4A108	**AXIMC_M8_FN_MOD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN_MOD[1:0]	
	Reset value																															0	0
0x4A10C - 0x4B0FC	Reserved	Reserved																															
0x4B100	**AXIMC_M9_READ_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													1	0	1	1
0x4B104	**AXIMC_M9_WRITE_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													1	0	1	1
0x4B108	**AXIMC_M9_FN_MOD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0
0x4B10C - 0x4C0FC	Reserved	Reserved																															
0x4C100	**AXIMC_M10_READ_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR_QOS[3:0]			
	Reset value																													0	0	1	0
0x4C104	**AXIMC_M10_WRITE_QOS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AW_QOS[3:0]			
	Reset value																													0	0	1	0
0x4C108	**AXIMC_M10_FN_MOD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRTIE_ISS_OVERRIDE	READ_ISS_OVERRIDE
	Reset value																															0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

2.5 Memory organization

2.5.1 Introduction

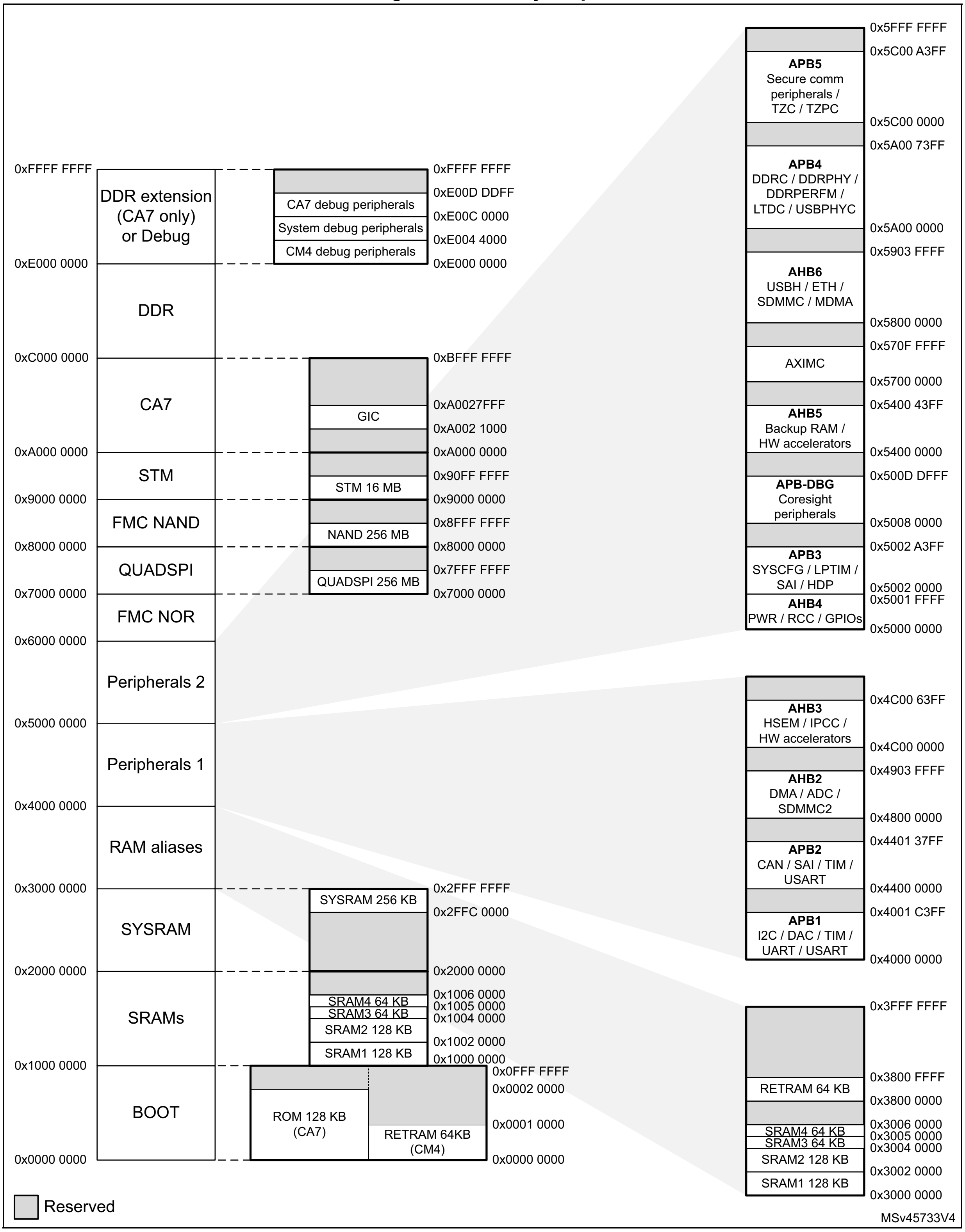
Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word’s least significant byte and the highest numbered byte the most significant.

The access to some part of the address space is master dependent, see *Section 2: Memory and bus architecture* for details.

2.5.2 Memory map and register boundary addresses

Figure 4. Memory map



All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and register areas, refer to the following table.

The following table gives the boundary addresses of the peripherals available in the devices.

Peripheral mapping

Table 9. Register boundary addresses

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
Cortex-A7 internal	0xA0026000 - 0xA0027FFF	8 KB	GICV	GIC virtual CPU interface (GICV)
	0xA0024000 - 0xA0025FFF	8 KB	GICH	GIC virtual interface control, common (GICH)
	0xA0022000 - 0xA0023FFF	8 KB	GICC	GIC CPU Interface (GICC)
	0xA0021000 - 0xA0021FFF	4 KB	GICD	GIC distributor (GICD)
	0xA0000000 - 0xA0020FFF	132 KB	Reserved	-
APB5	0x5C00A400 - 0x5FFFFFFF	65495 KB	Reserved	-
	0x5C00A000 - 0x5C00A3FF	1 KB	TAMP	TAMP registers
	0x5C009400 - 0x5C009FFF	3 KB	Reserved	-
	0x5C009000 - 0x5C0093FF	1 KB	I2C6	I2C registers
	0x5C008000 - 0x5C008FFF	4 KB	STGENC	STGEN registers
	0x5C007400 - 0x5C007FFF	3 KB	Reserved	-
	0x5C007000 - 0x5C0073FF	1 KB	ETZPC	ETZPC registers
	0x5C006000 - 0x5C006FFF	4 KB	TZC	TZC registers
	0x5C005000 - 0x5C005FFF	4 KB	BSEC	BSEC registers
	0x5C004400 - 0x5C004FFF	3 KB	Reserved	-
	0x5C004000 - 0x5C0043FF	1 KB	RTC	RTC registers
	0x5C003400 - 0x5C003FFF	3 KB	Reserved	-
	0x5C003000 - 0x5C0033FF	1 KB	IWDG1	IWDG registers
	0x5C002400 - 0x5C002FFF	3 KB	Reserved	-
	0x5C002000 - 0x5C0023FF	1 KB	I2C4	I2C registers
	0x5C001400 - 0x5C001FFF	3 KB	Reserved	-
	0x5C001000 - 0x5C0013FF	1 KB	SPI6	SPI/I2S registers
	0x5C000400 - 0x5C000FFF	3 KB	Reserved	-
	0x5C000000 - 0x5C0003FF	1 KB	USART1	USART registers

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
APB4	0x5A007400 - 0x5BFFFFFF	32739 KB	Reserved	-
	0x5A007000 - 0x5A0073FF	1 KB	DDRPERFM	*DDRPERFM registers*
	0x5A006000 - 0x5A006FFF	4 KB	USBPHYC	*USBPHYC registers*
	0x5A005000 - 0x5A005FFF	4 KB	STGENR	*STGEN registers*
	0x5A004000 - 0x5A004FFF	4 KB	DDRPHYC	*PUBL registers*
	0x5A003000 - 0x5A003FFF	4 KB	DDRCTRL	*DDRCTRL registers*
	0x5A002400 - 0x5A002FFF	3 KB	Reserved	-
	0x5A002000 - 0x5A0023FF	1 KB	IWDG2	*IWDG registers*
	0x5A001400 - 0x5A001FFF	3 KB	Reserved	-
	0x5A001000 - 0x5A0013FF	1 KB	LTDC	*LTDC registers*
	0x5A000000 - 0x5A000FFF	4 KB	Reserved	-
AHB6	0x5800E000 - 0x59FFFFFF	32712 KB	Reserved	-
	0x5800D000 - 0x5800DFFF	4 KB	USBH_EHCI	*USBH registers*
	0x5800C000 - 0x5800CFFF	4 KB	USBH_OHCI	*USBH registers*
	0x5800A000 - 0x5800BFFF	8 KB	ETH1	*Ethernet registers*
	0x58009000 - 0x58009FFF	4 KB	CRC1	*CRC registers*
	0x58008000 - 0x58008FFF	4 KB	DLYBSD2	*DLYB registers*
	0x58007000 - 0x58007FFF	4 KB	SDMMC2	*SDMMC registers*
	0x58006000 - 0x58006FFF	4 KB	DLYBSD1	*DLYB registers*
	0x58005000 - 0x58005FFF	4 KB	SDMMC1	*SDMMC registers*
	0x58004000 - 0x58004FFF	4 KB	DLYBQS	*DLYB registers*
	0x58003000 - 0x58003FFF	4 KB	QUADSPI	*QUADSPI registers*
	0x58002000 - 0x58002FFF	4 KB	FMC	*NOR/PSRAM controller registers* *NAND flash controller registers*
	0x58001000 - 0x58001FFF	4 KB	Reserved	-
	0x58000000 - 0x58000FFF	4 KB	MDMA	*MDMA registers*
AXIM	0x57100000 - 0x57FFFFFF	15360 KB	Reserved	-
	0x57000000 - 0x570FFFFF	1024 KB	AXIMC	*AXIMC registers*

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
AHB5	0x54004400 - 0x56FFFFFF	49135 KB	Reserved	-
	0x54004000 - 0x540043FF	1 KB	GPIOZ	*GPIO registers*
	0x54003400 - 0x54003FFF	3 KB	Reserved	-
	0x54003000 - 0x540033FF	1 KB	RNG1	*RNG registers*
	0x54002400 - 0x54002FFF	3 KB	Reserved	-
	0x54002000 - 0x540023FF	1 KB	HASH1	*HASH registers*
	0x54001400 - 0x54001FFF	3 KB	Reserved	-
	0x54001000 - 0x540013FF	1 KB	CRYP1(1)	*CRYP registers*
	0x54000000 - 0x54000FFF	4 KB	BKPSRAM	-
APB3	0x5002A400 - 0x5007FFFF	343 KB	Reserved	-
	0x5002A000 - 0x5002A3FF	1 KB	HDP	*HDP registers*
	0x50028400 - 0x50029FFF	7 KB	Reserved	-
	0x50028000 - 0x500283FF	1 KB	DTS	*DTS registers*
	0x50027400 - 0x50027FFF	3 KB	Reserved	-
	0x50027000 - 0x500273FF	1 KB	SAI4	*SAI registers*
	0x50025400 - 0x50026FFF	7 KB	Reserved	-
	0x50025000 - 0x500253FF	1 KB	VREFBUF	*VREFBUF registers*
	0x50024400 - 0x50024FFF	3 KB	Reserved	-
	0x50024000 - 0x500243FF	1 KB	LPTIM5	*LPTIM registers*
	0x50023400 - 0x50023FFF	3 KB	Reserved	-
	0x50023000 - 0x500233FF	1 KB	LPTIM4	*LPTIM registers*
	0x50022400 - 0x50022FFF	3 KB	Reserved	-
	0x50022000 - 0x500223FF	1 KB	LPTIM3	*LPTIM registers*
	0x50021400 - 0x50021FFF	3 KB	Reserved	-
	0x50021000 - 0x500213FF	1 KB	LPTIM2	*LPTIM registers*
	0x50020400 - 0x50020FFF	3 KB	Reserved	-
	0x50020000 - 0x500203FF	1 KB	SYSCFG	*SYSCFG registers*

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
AHB4	0x5000D400 - 0x5001FFFF	75 KB	Reserved	-
	0x5000D000 - 0x5000D3FF	1 KB	EXTI	*EXTI registers*
	0x5000C400 - 0x5000CFFF	3 KB	Reserved	-
	0x5000C000 - 0x5000C3FF	1 KB	GPIOK	*GPIO registers*
	0x5000B400 - 0x5000BFFF	3 KB	Reserved	-
	0x5000B000 - 0x5000B3FF	1 KB	GPIOJ	*GPIO registers*
	0x5000A400 - 0x5000AFFF	3 KB	Reserved	-
	0x5000A000 - 0x5000A3FF	1 KB	GPIOI	*GPIO registers*
	0x50009400 - 0x50009FFF	3 KB	Reserved	-
	0x50009000 - 0x500093FF	1 KB	GPIOH	*GPIO registers*
	0x50008400 - 0x50008FFF	3 KB	Reserved	-
	0x50008000 - 0x500083FF	1 KB	GPIOG	*GPIO registers*
	0x50007400 - 0x50007FFF	3 KB	Reserved	-
	0x50007000 - 0x500073FF	1 KB	GPIOF	*GPIO registers*
	0x50006400 - 0x50006FFF	3 KB	Reserved	-
	0x50006000 - 0x500063FF	1 KB	GPIOE	*GPIO registers*
	0x50005400 - 0x50005FFF	3 KB	Reserved	-
	0x50005000 - 0x500053FF	1 KB	GPIOD	*GPIO registers*
	0x50004400 - 0x50004FFF	3 KB	Reserved	-
	0x50004000 - 0x500043FF	1 KB	GPIOC	*GPIO registers*
	0x50003400 - 0x50003FFF	3 KB	Reserved	-
	0x50003000 - 0x500033FF	1 KB	GPIOB	*GPIO registers*
	0x50002400 - 0x50002FFF	3 KB	Reserved	-
	0x50002000 - 0x500023FF	1 KB	GPIOA	*GPIO registers*
	0x50001400 - 0x50001FFF	3 KB	Reserved	-
	0x50001000 - 0x500013FF	1 KB	PWR	*PWR registers*
	0x50000000 - 0x50000FFF	4 KB	RCC	*RCC registers*

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
AHB3	0x4C006400 - 0x4FFFFFFF	65511 KB	Reserved	-
	0x4C006000 - 0x4C0063FF	1 KB	DCMI	*DCMI registers*
	0x4C005400 - 0x4C005FFF	3 KB	Reserved	-
	0x4C005000 - 0x4C0053FF	1 KB	CRYP2(1)	*CRYP registers*
	0x4C004400 - 0x4C004FFF	3 KB	Reserved	-
	0x4C004000 - 0x4C0043FF	1 KB	CRC2	*CRC registers*
	0x4C003400 - 0x4C003FFF	3 KB	Reserved	-
	0x4C003000 - 0x4C0033FF	1 KB	RNG2	*RNG registers*
	0x4C002400 - 0x4C002FFF	3 KB	Reserved	-
	0x4C002000 - 0x4C0023FF	1 KB	HASH2	*HASH registers*
	0x4C001400 - 0x4C001FFF	3 KB	Reserved	-
	0x4C001000 - 0x4C0013FF	1 KB	IPCC	*IPCC registers*
	0x4C000400 - 0x4C000FFF	3 KB	Reserved	-
	0x4C000000 - 0x4C0003FF	1 KB	HSEM	*HSEM registers*
AHB2	0x49040000 - 0x4BFFFFFF	48896 KB	Reserved	-
	0x49000000 - 0x4903FFFF	256 KB	OTG	*OTG registers*
	0x48005400 - 0x48FFFFFF	16363 KB	Reserved	-
	0x48005000 - 0x480053FF	1 KB	DLYBSD3	*DLYB registers*
	0x48004400 - 0x48004FFF	3 KB	Reserved	-
	0x48004000 - 0x480043FF	1 KB	SDMMC3	*SDMMC registers*
	0x48003400 - 0x48003FFF	3 KB	Reserved	-
	0x48003000 - 0x480033FF	1 KB	ADC12	*ADC registers (for each ADC)*
	0x48002400 - 0x48002FFF	3 KB	Reserved	-
	0x48002000 - 0x480023FF	1 KB	DMAMUX1	*DMAMUX registers*
	0x48001400 - 0x48001FFF	3 KB	Reserved	-
	0x48001000 - 0x480013FF	1 KB	DMA2	*DMA registers*
	0x48000400 - 0x48000FFF	3 KB	Reserved	-
	0x48000000 - 0x480003FF	1 KB	DMA1	*DMA registers*

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
APB2	0x44013800 - 0x47FFFFFF	65458 KB	Reserved	-
	0x44011000 - 0x440137FF	10 KB	CANSRAM	-
	0x44010400 - 0x44010FFF	3 KB	Reserved	-
	0x44010000 - 0x440103FF	1 KB	CCU	*CCU registers*
	0x4400F400 - 0x4400FFFF	3 KB	Reserved	-
	0x4400F000 - 0x4400F3FF	1 KB	FDCAN2	*FDCAN registers*
	0x4400E400 - 0x4400EFFF	3 KB	Reserved	-
	0x4400E000 - 0x4400E3FF	1 KB	FDCAN1	*FDCAN registers*
	0x4400D800 - 0x4400DFFF	2 KB	Reserved	-
	0x4400D000 - 0x4400D7FF	2 KB	DFSDM1	*DFSDM channel y registers (y=0..7)* *DFSDM filter x module registers (x=0..5)*
	0x4400C400 - 0x4400CFFF	3 KB	Reserved	-
	0x4400C000 - 0x4400C3FF	1 KB	SAI3	*SAI registers*
	0x4400B400 - 0x4400BFFF	3 KB	Reserved	-
	0x4400B000 - 0x4400B3FF	1 KB	SAI2	*SAI registers*
	0x4400A400 - 0x4400AFFF	3 KB	Reserved	-
	0x4400A000 - 0x4400A3FF	1 KB	SAI1	*SAI registers*
	0x44009400 - 0x44009FFF	3 KB	Reserved	-
	0x44009000 - 0x440093FF	1 KB	SPI5	*SPI/I2S registers*
	0x44008400 - 0x44008FFF	3 KB	Reserved	-
	0x44008000 - 0x440083FF	1 KB	TIM17	*TIM16/TIM17 registers*
	0x44007400 - 0x44007FFF	3 KB	Reserved	-
	0x44007000 - 0x440073FF	1 KB	TIM16	*TIM16/TIM17 registers*
	0x44006400 - 0x44006FFF	3 KB	Reserved	-
	0x44006000 - 0x440063FF	1 KB	TIM15	*TIM15 registers*
	0x44005400 - 0x44005FFF	3 KB	Reserved	-
	0x44005000 - 0x440053FF	1 KB	SPI4	*SPI/I2S registers*
	0x44004400 - 0x44004FFF	3 KB	Reserved	-
	0x44004000 - 0x440043FF	1 KB	SPI1	*SPI/I2S registers*
	0x44003400 - 0x44003FFF	3 KB	Reserved	-
	0x44003000 - 0x440033FF	1 KB	USART6	*USART registers*
	0x44001400 - 0x44002FFF	7 KB	Reserved	-
	0x44001000 - 0x440013FF	1 KB	TIM 8	*TIM1/TIM8 registers*
	0x44000400 - 0x44000FFF	3 KB	Reserved	-
	0x44000000 - 0x440003FF	1 KB	TIM 1	*TIM1/TIM8 registers*

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
APB1	0x4001C400 - 0x43FFFFFF	65423 KB	Reserved	-
	0x4001C000 - 0x4001C3FF	1 KB	MDIOS	*MDIOS registers*
	0x40019400 - 0x4001BFFF	11 KB	Reserved	-
	0x40019000 - 0x400193FF	1 KB	UART8	*USART registers*
	0x40018400 - 0x40018FFF	3 KB	Reserved	-
	0x40018000 - 0x400183FF	1 KB	UART7	*USART registers*
	0x40017400 - 0x40017FFF	3 KB	Reserved	-
	0x40017000 - 0x400173FF	1 KB	DAC1	*DAC registers*
	0x40016400 - 0x40016FFF	3 KB	Reserved	-
	0x40016000 - 0x400163FF	1 KB	CEC	*HDMI-CEC registers*
	0x40015400 - 0x40015FFF	3 KB	Reserved	-
	0x40015000 - 0x400153FF	1 KB	I2C5	*I2C registers*
	0x40014400 - 0x40014FFF	3 KB	Reserved	-
	0x40014000 - 0x400143FF	1 KB	I2C3	*I2C registers*
	0x40013400 - 0x40013FFF	3 KB	Reserved	-
	0x40013000 - 0x400133FF	1 KB	I2C2	*I2C registers*
	0x40012400 - 0x40012FFF	3 KB	Reserved	-
	0x40012000 - 0x400123FF	1 KB	I2C1	*I2C registers*
	0x40011400 - 0x40011FFF	3 KB	Reserved	-
	0x40011000 - 0x400113FF	1 KB	UART5	*USART registers*
	0x40010400 - 0x40010FFF	3 KB	Reserved	-
	0x40010000 - 0x400103FF	1 KB	UART4	*USART registers*

Table 9. Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral	Peripheral Register map
APB1	0x4000F400 - 0x4000FFFF	3 KB	Reserved	-
	0x4000F000 - 0x4000F3FF	1 KB	USART3	*USART registers*
	0x4000E400 - 0x4000EFFF	3 KB	Reserved	-
	0x4000E000 - 0x4000E3FF	1 KB	USART2	*USART registers*
	0x4000D400 - 0x4000DFFF	3 KB	Reserved	-
	0x4000D000 - 0x4000D3FF	1 KB	SPDIFRX	*SPDIFRX interface registers*
	0x4000C400 - 0x4000CFFF	3 KB	Reserved	-
	0x4000C000 - 0x4000C3FF	1 KB	SPI3	*SPI/I2S registers*
	0x4000B400 - 0x4000BFFF	3 KB	Reserved	-
	0x4000B000 - 0x4000B3FF	1 KB	SPI2	*SPI/I2S registers*
	0x4000A400 - 0x4000AFFF	3 KB	Reserved	-
	0x4000A000 - 0x4000A3FF	1 KB	WWDG1	*WWDG registers*
	0x40009400 - 0x40009FFF	3 KB	Reserved	-
	0x40009000 - 0x400093FF	1 KB	LPTIM1	*LPTIM registers*
	0x40008400 - 0x40008FFF	3 KB	Reserved	-
	0x40008000 - 0x400083FF	1 KB	TIM14	*TIM13/TIM14 registers*
	0x40007400 - 0x40007FFF	3 KB	Reserved	-
	0x40007000 - 0x400073FF	1 KB	TIM13	*TIM13/TIM14 registers*
	0x40006400 - 0x40006FFF	3 KB	Reserved	-
	0x40006000 - 0x400063FF	1 KB	TIM12	*TIM12 registers*
	0x40005400 - 0x40005FFF	3 KB	Reserved	-
	0x40005000 - 0x400053FF	1 KB	TIM7	*TIM6/TIM7 registers*
	0x40004400 - 0x40004FFF	3 KB	Reserved	-
	0x40004000 - 0x400043FF	1 KB	TIM6	*TIM6/TIM7 registers*
	0x40003400 - 0x40003FFF	3 KB	Reserved	-
	0x40003000 - 0x400033FF	1 KB	TIM5	*TIM2/TIM3/TIM4/TIM5 registers*
	0x40002400 - 0x40002FFF	3 KB	Reserved	-
	0x40002000 - 0x400023FF	1 KB	TIM4	*TIM2/TIM3/TIM4/TIM5 registers*
	0x40001400 - 0x40001FFF	3 KB	Reserved	-
	0x40001000 - 0x400013FF	1 KB	TIM3	*TIM2/TIM3/TIM4/TIM5 registers*
	0x40000400 - 0x40000FFF	3 KB	Reserved	-
	0x40000000 - 0x400003FF	1 KB	TIM2	*TIM2/TIM3/TIM4/TIM5 registers*

1. Check device datasheet for availability.

Debugger mapping

Table 10. Debugger Register boundary addresses

Bus	Boundary address	Size (Bytes)	Peripheral
DAPBUS (1)	0xE0100000 - 0xFFFFFFFF	523264 KB	Reserved
	0xE00FF000 - 0xE00FFFFF	4 KB	CM4 ROM Table(2)
	0xE00DE000 - 0xE00FEFFF	132 KB	Reserved
	0xE00DD000 - 0xE00DDFFF	4 KB	CA7_ETM1
	0xE00DC000 - 0xE00DCFFF	4 KB	CA7_ETM0
	0xE00DA000 - 0xE00DBFFF	8 KB	Reserved
	0xE00D9000 - 0xE00D9FFF	4 KB	CA7_CTI1
	0xE00D8000 - 0xE00D8FFF	4 KB	CA7_CTI0
	0xE00D4000 - 0xE00D7FFF	16 KB	Reserved
	0xE00D3000 - 0xE00D3FFF	4 KB	CA7_PMU1
	0xE00D2000 - 0xE00D2FFF	4 KB	CA7_DBG1
	0xE00D1000 - 0xE00D1FFF	4 KB	CA7_PMU0
	0xE00D0000 - 0xE00D0FFF	4 KB	CA7_DBG0
	0xE00C1000 - 0xE00CFFFF	60 KB	Reserved
	0xE00C0000 - 0xE00C0FFF	4 KB	CA7_ROM
	0xE00A1000 - 0xE00BFFFF	124 KB	Reserved
	0xE00A0000 - 0xE00A0FFF	4 KB	*STM registers*
	0xE0095000 - 0xE009FFFF	44 KB	Reserved
	0xE0094000 - 0xE0094FFF	4 KB	CTI
	0xE0093000 - 0xE0093FFF	4 KB	TPIU
	0xE0092000 - 0xE0092FFF	4 KB	ETF
	0xE0091000 - 0xE0091FFF	4 KB	Main Funnel
	0xE0090000 - 0xE0090FFF	4 KB	Trace Subsystem ROM Table
	0xE0084000 - 0xE008FFFF	48 KB	Reserved
	0xE0083000 - 0xE0083FFF	4 KB	SWO
	0xE0082000 - 0xE0082FFF	4 KB	TSGEN1 non-secure PORT
	0xE0081000 - 0xE0081FFF	4 KB	DBG_MCU
	0xE0080000 - 0xE0080FFF	4 KB	DAP ROM Table
	0xE0044000 - 0xE007FFFF	240 KB	Reserved
	0xE0043000 - 0xE0043FFF	4 KB	CM4 CTI(2)
	0xE0042000 - 0xE0042FFF	4 KB	Reserved
	0xE0041000 - 0xE0041FFF	4 KB	CM4 ETM(2)

Table 10. Debugger Register boundary addresses (continued)

Bus	Boundary address	Size (Bytes)	Peripheral
DAPBUS (1)	0xE000F000 - 0xE0040FFF	200 KB	Reserved
	0xE000E000 - 0xE000EFFF	4 KB	CM4 SCS(2)
	0xE0003000 - 0xE000DFFF	44 KB	Reserved
	0xE0002000 - 0xE0002FFF	4 KB	CM4 FPB(2)
	0xE0001000 - 0xE0001FFF	4 KB	CM4 DWT(2)
	0xE0000000 - 0xE0000FFF	4 KB	CM4 ITM(2)
Debug APB(3)	0x500DE000 - 0x53FFFFFF	64648 KB	Reserved
	0x500DD000 - 0x500DDFFF	4 KB	CA7_ETM1
	0x500DC000 - 0x500DCFFF	4 KB	CA7_ETM0
	0x500DA000 - 0x500DBFFF	8 KB	Reserved
	0x500D9000 - 0x500D9FFF	4 KB	CA7_CTI1
	0x500D8000 - 0x500D8FFF	4 KB	CA7_CTI0
	0x500D4000 - 0x500D7FFF	16 KB	Reserved
	0x500D3000 - 0x500D3FFF	4 KB	CA7_PMU1
	0x500D2000 - 0x500D2FFF	4 KB	CA7_DBG1
	0x500D1000 - 0x500D1FFF	4 KB	CA7_PMU0
	0x500D0000 - 0x500D0FFF	4 KB	CA7_DBG0
	0x500C1000 - 0x500CFFFF	60 KB	Reserved
	0x500C0000 - 0x500C0FFF	4 KB	CA7_ROM
	0x500A1000 - 0x500BFFFF	124 KB	Reserved
	0x500A0000 - 0x500A0FFF	4 KB	*STM registers*
	0x50095000 - 0x5009FFFF	44 KB	Reserved
	0x50094000 - 0x50094FFF	4 KB	CTI
	0x50093000 - 0x50093FFF	4 KB	TPIU
	0x50092000 - 0x50092FFF	4 KB	ETF
	0x50091000 - 0x50091FFF	4 KB	Funnel
	0x50090000 - 0x50090FFF	4 KB	Trace Subsystem ROM Table
	0x50084000 - 0x5008FFFF	48 KB	Reserved
	0x50083000 - 0x50083FFF	4 KB	SWO
	0x50082000 - 0x50082FFF	4 KB	TSGEN1
	0x50081000 - 0x50081FFF	4 KB	DBGMCU
	0x50080000 - 0x50080FFF	4 KB	DAP ROM Table

1. Accessible by debugger only.
2. Also accessible locally by Cortex®-M4.
3. Accessible by Debugger, Cortex®-M4 and Cortex®-A7

3 Boot and security and OTP control (BSEC)

3.1 Introduction

The BSEC (boot and security and OTP control) is intended to control an OTP (one time programmable) fuse box, used for on-chip non-volatile storage for device configuration and security parameters.

3.2 BSEC main features

- 32-bit APB4 interface
- 4096 OTP bits (with 3072 effective bits)
- Valid power-supply **pwrok** input so that OTP is only operated within a VDD/VDD1 valid range
- Scratch register for communication with external agent and to store boot parameters
- Feature locking mechanism: BSEC provides 15 hardware signals, controlled by OTP to disable specific SoC features for product differentiation
- JTAG SOC interface via BSEC_JTAGIN and BSEC_JTAGOUT registers for communication channel to JTAG TAP controller.
- Two OTP regions designated as lower OTP (1 Kbits) and upper OTP (2 Kbits) having different properties
- **Lower OTP bits** (1 Kbits), with **2:1 redundancy**, are intended to store:
  - Device configuration: BSEC OTP mode and device features
  - Boot parameters
  - Engineering data and analog trim
  - Public root key (256 bits)

  All basic security features, including secure Boot with code authentication, are supported **only** with lower OTP region.
- **Upper OTP bits** (2 Kbits) with 6 bits per 32-bit word ECC support, are intended to store security sensitive information
  - ST secret keys
  - OEM secrets

  Upper OTP should only be treated by 32-bit words
- Improved resistance to hardware attacks by clock and power glitches during OTP read
- OTP bits are always read with disturbcheck.

  Every OTP word read is internally compared to an internal reference level. The disturbcheck information is provided to BSEC to qualify the OTP word value.
- Global programming locking by sticky bits
- Permanent OTP program locking per word
- OTP word programming lock by sticky bits during Boot
- Shadow OTP registers can be individually write locked by sticky bits during Boot
- Shadow OTP registers can be individually read locked by sticky bits during Boot (to prevent reloading)
- Arm® TrustZone® awareness applicable to:

- BSEC control registers
- BSEC shadow registers from lower OTP region (words 0 to 31)
- BSEC shadow registers from upper OTP region (words 32 to 95)

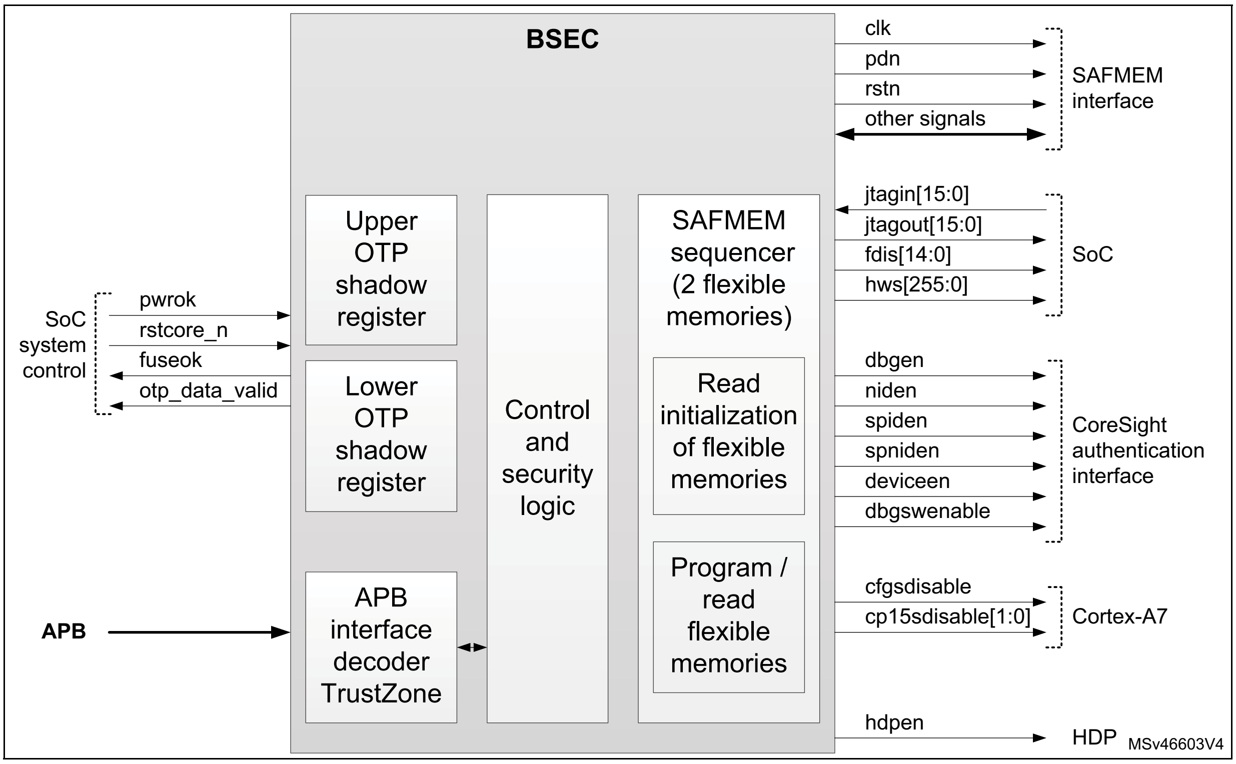
- Arm® CoreSight™ debug authentication interface with six signals:
  - **dbgen** (debug enable)
  - **devicen** (external debug enable)
  - **niden** (non-invasive debug enable)
  - **spiden** (secure privilege invasive debug enable)
  - **spniden** (secure privilege non-invasive debug enable)
  - **dbgswenable** (self hosted debug enable)
- Up to 256 hardware signals **hws[255:0]** for analog trim, factory settings and SoC RAM repair

3.3 BSEC functional description

BSEC provides an interface to the OTP (also known as anti-fuse macro) containing 3072 useful bits. BSEC also provides several registers used by the Boot process as well as mechanisms to prevent access to security sensitive information.

3.3.1 BSEC block diagram

Figure 5. BSEC block diagram



3.3.2 Interface to OTP

OTP bits are grouped in words of 32 bits each and are organized in two regions:

- the **lower region**: 32 words of 32 bits (lower 2 Kbits), using a 2:1 redundancy. SoC and Boot configuration parameters, engineering data and public root key hash are stored in this lower region.
- the **upper region**: 64 words of 32 bits (2 Kbits upper half), using ECC scheme (32 bits encoded into 38 bits). Security sensitive parameters such as secret keys are stored in this upper region.

Note: OTP bits are initially set to 0 and can be programmed once to one.
Lower OTP words, using 2:1 redundancy, can be bitwise programmed in several operations
Upper OTP words, using internal ECC, must be programmed in a single operation as a bulk 32 bits data.

The OTP bits are not directly accessible by the CPU. Instead the CPU must access shadow registers that contain copies of the OTP useful bits.

There are 96 **shadow registers: BSEC_OTP_DATA0 to BSEC_OTP_DATA95**.

These 32-bit registers are copies of actual OTP bits. Redundancy and ECC are made transparent to the user. BSEC_OTP_DATAx are also referred to OTP word #x in this document.

There are two possibilities to udpate the shadow registers:

- a system reset, where BSEC automatically copies the whole set of OTP useful bits to the shadow registers.
- an OTP read operation requested by the CPU, used after an OTP programming operation to update the corresponding shadow register.

The following OTP useful bits have a special meaning for BSEC:

- **word #0: bits 0 to 6** reserved to determine OTP security mode
- **word #1: bits 0 to 15** used to control feature disabling
- words #16 to word #23 are connected to the SoC signals hws[255:0].
  These signals are either:
  - connected to SoC peripherals (example: for analog peripheral factory trimming, or for memory repair).
  - left unconnected if OTP bits are only used by software (see *Section 4: OTP mapping (OTP)*).

3.3.3 OTP security modes

OTP word#0 bits 0 to 6 (with their potential disturbed state) are used to determine the OTP security mode.

The following OTP security modes are available:

- OTP-SECURED, including sub-modes, according to OTP word#0 as described in *Table 11*:
  - open_device
  - closed_device
- OTP-INVALID
  - In this state the upper OTP shadow registers are cleared and not accessible, the lower OTP shadow registers are not accessible, BSEC control interface is read/write accessible to secure only.

The OTP mode is made visible to software by register BSEC_OTP_STATUS.

Table 11. OTP modes definition

BSEC_OTP_DATA0 [6:0]	OTP mode	BSEC_OTP_STATUS SECURE	BSEC_OTP_STATUS INVALID
0xxx1x1	**OTP-SECURED open_device**	1	0
01x1xxx			
0xx11xx			
01xxxx1			
1xxx1x1	**OTP-SECURED closed_device**	1	0
11x1xxx			
1xx11xx			
11xxxx1			
All other values	**OTP-INVALID**	x	1

Note: The OTP mode is computed from BSEC_OTP_DATA0[6:0] only if BSEC_OTP_DISTURBED[0] is equal to zero. Otherwise the OTP mode is forced to OTP-INVALID.

3.3.4 Automatic OTP load on system-reset

On a system-reset, BSEC automatically updates all shadow registers. The OTP mode is determined during this phase.

For OTP-**SECURED** mode, all OTP words are read and BSEC_OTP_DATA**0** to BSEC_OTP_DATA**95** are updated.

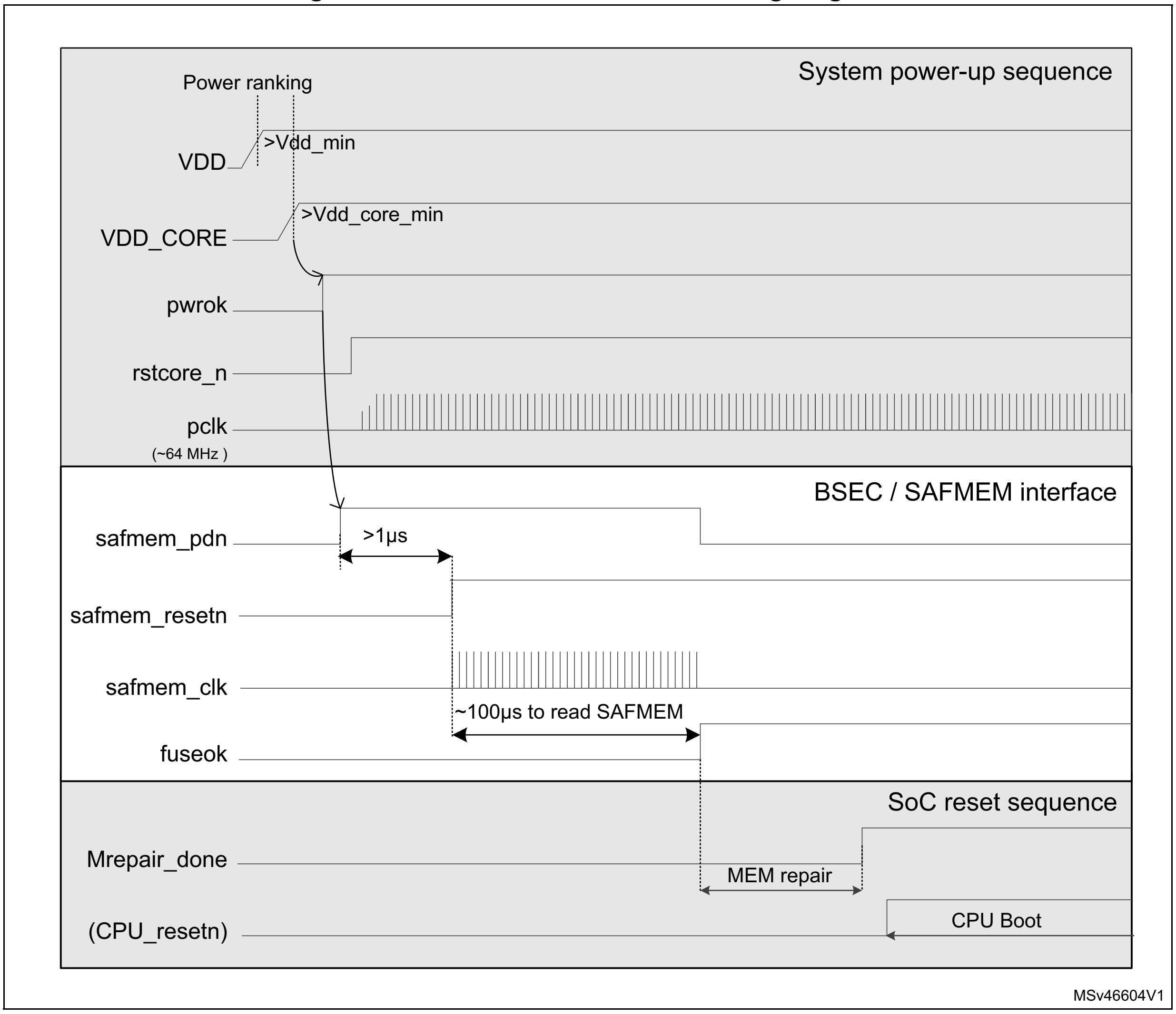
For OTP-**INVALID** mode, only BSEC_OTP_DATA**0** to BSEC_OTP_DATA**31** are updated. BSEC_OTP_DATA**32** to BSEC_OTP_DATA**95**
are set to 0.

BSEC_OTP_STATUS, BSEC_OTP_DISTURBEDx and BSEC_OTP_ERRORx registers are also updated.

fuseok signal is asserted at the end of this phase. This signal is used to release the reset to the SoC.

Note: the OTP word reading starts from word 0, in order for the BSEC to determine the OTP mode from the beginning.

Figure 6. Power-on and initial read timing diagram



3.3.5 OTP words status

- **BSEC_OTP_DISTURBEDx** registers (x = 0...2) provide the **disturbed status** for each OTP word. Any bit set to 1 means that the last reading of the corresponding word has been disturbed. Abnormal reading conditions in circuitry decoding and voltages reading have been detected.
- **BSEC_OTP_ERRORx** registers (x = 0...2) provide **error status** for each OTP word. Any bit set to 1 means that the last read operation of the word concerned revealed a redundancy or ECC check error.
- **BSEC_OTP_WRLOCKEDx** registers (x = 0...2) provide a **permanent write lock** status for each OTP word. Any bit set to 1 means that the corresponding word is permanently locked for writing.

Note: All status registers are updated during the initial read, but the disturbed and error status registers are also updated after each OTP read operation.
The first 32 OTP words implement a scheme to improve OTP robustness against hardware attacks during read as follows: Spare bits 32 to 37 are intended to support ECC and are not

used by 2:1 redundancy. They are therefore written with a simple pattern (for example OTP word address on 5 bits with even parity on the 6th bit).This pattern is checked during every OTP read. If the value read on bit 32 to 37 does not match the expected value, a disturbed information is reported to BSEC_OTP_DISTURBED0 register.

3.3.6 OTP operations

BSEC is clocked by PCLK clock source (single clock for functional and interface registers). An OTP word reading, writing (programming) or locking is called an OTP operation.

One OTP operation is performed via the BSEC_OTP_CONTROL register. It starts when the software writes a valid request to BSEC_OTP_CONTROL register and when no other operation is on going (an OTP operation cannot start until the previous one is finished).

Specific read, programming or permanent write lock operations are detailed below.

OTP read

A read operation consists in updating one of BSEC_OTP_DATA0 to BSEC_OTP_DATA95 registers with the corresponding word from OTP word.

To trigger a read operation, the software must set BSEC_OTP_CONTROL register with the word number given in ADDR field and with PROG bit set to 0.

The software can check the BUSY bit from the BSEC_OTP_STATUS register: Once cleared, this BUSY bit indicates that the read operation is complete.

When the read operation is finished, the BSEC state machine updates the “disturbed” and “error” status registers.

BSEC parameters depending on OTP content, are also updated when the corresponding OTP words are read. The OTP mode is updated when a read operation of the word 0 is performed.

Note: In OTP-INVALID mode, or if bit OTP in BSEC_OTP_LOCK register is set, words 32 to 95 cannot be read. The corresponding shadow registers are not modified.

OTP programming

A programming operation consists in programming one or several bits of an OTP word to one.

OTP word bits may be incrementally programmed to one but bits programmed to one can not be written to zero.

Lower OTP words are bitwise programmable and can be programmed incrementally to one's, upper OTP words have to be programmed in a single operation because of ECC encoding.

To trigger a programming operation, the two following steps are required:

- set the word value to write in the BSEC_OTP_WRDATA register
- set the BSEC_OTP_CONTROL register with:
  - word number given in ADDR field
  - PROG bit set to 1
  - LOCK bit set to 0.

The software checks then the BUSY bit from the BSEC_OTP_STATUS register: Once cleared, this BUSY bit indicates that the write operation is complete.

In the same register, the progfail bit is set if the write operation has failed.

Note: a programming operation to an OTP word does not update the corresponding shadow register. If needed, the software can update the shadow register with a read operation. In OTP-INVALID mode, the upper OTP (words 32 to 95) can not be programmed, regardless of bit GPLOCK in BSEC_OTP_LOCK register and regardless of registers BSEC_OTP_SPLOCK2/3.

OTP permanent write lock

A permanent write lock operation consists in forcing an OTP word to be permanently write-locked and not programmable.

For permanent write lock of an OTP word, the software must perform the same sequence as a normal programming operation with bit LOCK in BSEC_OTP_CONTROL register, with field ADDR in BSEC_OTP_CONTROL register and with BSEC_OTP_WRDATA register (see *Table 12*). After permanent lock programming, it is required to update BSEC_OTP_WRLOCKEDx registers with OTP content. In order to do this, the software must perform a word read with LOCK bit set and word address provided in BSEC_OTP_CONTROL register.

Table 12. OTP permanent write lock

OTP word to be locked	BSEC control register value		Comment
	ADDR[6:0]	WRDATA[31:0]	
0	0x00	0x0000 0003	Use 2 bits per word for lower OTPs
1	0x00	0x0000 000C	
...	...	...	
7	0x00	0x0000 C000	
8	0x01	0x0000 0003	
...	...	...	
15	0x01	0x0000 C000	
16	0x02	0x0000 0003	
...	...	...	
31	0x03	0x0000 C000	
32	0x04	0x0000 0001	Use 1 bit per word for upper OTPs
33	0x04	0x0000 0002	
...	...	...	
47	0x04	0x0000 8000	
48	0x05	0x0000 0001	
...	...	...	
63	0x05	0x0000 8000	
...	...	...	
95	0x07	0x0000 8000	

3.3.7 JTAG registers

The BSEC_JTAGIN and BSEC_JTAGOUT registers are two 16-bit registers used to communicate between software and JTAG SoC interface. This interface is used during Boot as communication channel to the external JTAG TAP controller.

3.3.8 BSEC lock registers

BSEC_OTP_LOCK register is used to lock access to security sensitive areas with sticky bits. The locks are active until the next system-reset (see *Section 3.6.5: BSEC OTP lock configuration register (BSEC_OTP_LOCK)*).

3.3.9 Debug control

The software controls the scope of the Debug by BSEC_DENABLE register. The access to this register can be locked until next system-reset with bit DENREG in BSEC_OTP_LOCK register.

BSEC_DENABLE is driving hardware signals to the SoC (such as CoreSight authentication interface and other specific control listed in *Table 13*).

For more information, see *Section 64: Debug support (DBG)*.

BSEC_DENABLE default values depends on BSEC OTP mode as shown in *Table 13*.

Table 13. BSEC_DENABLE default values after reset

Signal	OTP-SECURED open_device	OTP-SECURED closed_device	OTP-INVALID	Comment
dbgen	1	0	0	CoreSight authentication
niden	1	0	0	CoreSight authentication
spiden	0	0	0	CoreSight authentication
spniden	0	0	0	CoreSight authentication
deviceen	1	0	0	CoreSight authentication
dbgswenable	0	0	0	CoreSight authentication
hdpen	0	0	0	Hardware debug port tracing
cfgsdisable	0	0	0	Disable some of Cortex®-A7 GIC secure access
cp15sdisable[1:0]	0b00	0b00	0b00	Disable some of Cortex®-A7 CP15 secure access

3.3.10 Sticky lock register

Three sticky lock registers are used to control the reload of shadow registers, at OTP word granularity:

- BSEC_SRLOCKx (x = 0..2) to prevent reloading of selected shadow registers until next system-reset.
- BSEC_SWLOCKx (x = 0..2) to lock the write to selected shadow register from APB until next system-reset.
- BSEC_SPLOCKx (x = 0..2) to prevent programming of selected OTP words until next system-reset.

3.4 OTP layout

Lower OTP words (1 Kbits)

The lower OTP bits are stored in shadow registers BSEC_OTP_DATA0 to BSEC_OTP_DATA31 and used for general purpose and non-secret data.

These OTP are protected by a 2:1 redundancy implemented with consecutive even/odd bits.

The specific bits impacting hardware such as OTP modes, engineering, trim and memory repair are located in: CFG0, CFG1 and 8 other words (HW0 to HW7).

Upper OTP words (2 Kbits)

The upper OTP words are stored in shadow registers BSEC_OTP_DATA32 to BSEC_OTP_DATA95 and are reserved to store security sensitive information (secrets keys, certificate, RMA key). These words may also store public non-volatile information.

These upper OTP bits can be used during the Boot process to support advanced security services such as secure secret provisioning (SSP), encrypted Boot or secure storage.

The upper OTP words are protected by ECC and must be used with 32-bit word granularity.

3.4.1 DMA requests

BSEC does not support DMA hardware interface.

3.4.2 TrustZone awareness

BSEC is TrustZone aware. All APB register read and write accesses are checked for permission. *Table 14* and *Table 15* describe the permission rules depending on OTP mode and register type.

If permission check fails:

- write access is ignored and SLVERR is signaled on APB bus.
- read access returns zeros and SLVERR is signaled on APB bus.

Table 14. Write access permissions

OTP mode	Control registers[1]		Lower OTP shadow registers		Upper OTP shadow registers	
	APB secure	APB non-secure	APB secure	APB non-secure	APB secure	APB non-secure
OTP-SECURED	Yes	No	Yes	No	Yes	No
OTP-INVALID	No	No	No	No	No	No

1. BSEC control registers are any BSEC registers except BSEC_OTP_DATA0 to BSEC_OTP_DATA95.

Table 15. Read access permissions

OTP mode	Control registers[1]		Lower OTP shadow registers		Upper OTP shadow registers	
	APB secure	APB non-secure	APB secure	APB non-secure	APB secure	APB non-secure
OTP-SECURED	Yes	Yes	Yes	Yes	Yes	No
OTP-INVALID	Yes	No	No	No	No	No

1. BSEC control registers are any BSEC registers except BSEC_OTP_DATA0 to BSEC_OTP_DATA95.

3.4.3 BSEC clocking and initialization

BSEC has a single clock domain (PCLK).

- During read and programming OTPs, PCLK frequency must be ≤ 67 MHz.
- During write to shadow registers, PCLK frequency must be ≤ 67 MHz.
- During all other operations, PCLK frequency is not restricted but it is recommended to keep PCLK ≤ 67 MHz for all operations (to avoid frequency changes).
- BSEC is enforcing the correct sequencing wrt signals: CLK, PDN, and RESETN during power-up and power-down sequences.

3.5 BSEC interrupts

BSEC does **not** support any interrupt.

3.6 BSEC registers

The peripheral registers are accessed by words (32-bit).

Any write access to read-only registers are silently ignored.

Any attempt to modify locked registers (locked by sticky locks or by lock bits) are silently ignored.

3.6.1 BSEC OTP configuration register (BSEC_OTP_CONFIG)

Address offset: 0x000

Reset value: 0x0000 000E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREAD[1:0]		PRGWIDTH[3:0]				FRC[1:0]		PWRUP
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:7 **TREAD[1:0]**: set OTP reading current level (default = 0b00)

Bits 6:3 **PRGWIDTH[3:0]**: OTP programming pulse width (default = 0b0001)

Bits 2:1 **FRC[1:0]**: OTP clock frequency range selection

00: 10 MHz ≤ frequency ≤ 20 MHz
01: 20 MHz ≤ frequency ≤ 30 MHz
10: 30 MHz ≤ frequency ≤ 45 MHz
11: 45 MHz ≤ frequency ≤ 67 MHz

Note: 0b11 is used by default. OTP *initial read is always using an internal OSC with 64 MHz max frequency. Reading* OTP *at a lower frequency than the range programmed by FRC[1:0] is possible.*

Bit 0 **PWRUP**: OTP power-up control

0: OTP powered down
1: OTP powered up

Note: After the power-on initial read of OTP*, BSEC powers down* OTP*. PWRUP bit is then cleared.*
Prior to any read or programming operation, OTP *must be powered up again by setting PWRUP bit.*
OTP *read is qualified by **pwrok** input signal, which indicates that VDD and VDD1 supplies are in valid range.*

3.6.2 BSEC OTP control register (BSEC_OTP_CONTROL)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	LOCK	PROG	Res.	ADDR[6:0]						
						rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **LOCK**: OTP permanent word lock control
0: OTP normal word programming/read
1: OTP permanent write lock word programming/read

Bit 8 **PROG**: OTP operation control
0: OTP read operation
1: OTP program operation

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **ADDR[6:0]**: OTP word address
Actual OTP word address.
Words address are from 0 to 95 for BSEC
Note: the actual OTP *word address is adjusted according to the redundancy or ECC scheme.*

3.6.3 BSEC OTP write data register (BSEC_OTP_WRDATA)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRDATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRDATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **WRDATA[31:0]**: OTP write data

3.6.4 BSEC OTP status register (BSEC_OTP_STATUS)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWR ON	PROG FAIL	BUSY	IN VALID	Res.	SEC URE
										r	r	r	r		r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 Reserved, must be kept at reset value.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PWRON**: OTP power status
0: OTP in power off
1: OTP in power on
Note: used to poll pwrok signal value

Bit 4 **PROGFAIL**: last programming status
0: OTP successful last programming
1: OTP failed last programming

Bit 3 **BUSY**: OTP operation status
0: OTP idle
1: OTP operation on going
Note: bit polling is used to determine operation completion.

Bit 2 **INVALID**: OTP invalid mode
0: OTP mode is not OTP-INVALID
1: OTP mode is OTP-INVALID

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SECURE**: OTP secured mode
0: OTP mode is not OTP-SECURED
1: OTP mode is OTP-SECURED

Note: The reset value of SECURE and INVALID bits depends on OTP mode.

Note: The lowest bits SECURE and INVALID must be used to identify the OTP mode instead of reading shadow register BSEC_OTP_DATA0. Precedence is INVALID > SECURE.

3.6.5 BSEC OTP lock configuration register (BSEC_OTP_LOCK)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GP LOCK	Res.	DEN REG	ROM LOCK	OTP
											rs		rs	rs	rs

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **GPLOCK**: programming sticky lock
0: programming allowed
1: programming disabled until next system reset

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DENREG**: debug enable register sticky lock
0: BSEC_DENABLE register not locked
1: BSEC_DENABLE register locked until next system reset

Bit 1 **ROMLOCK**: Upper ROM region read lock
0: Upper ROM region is accessible
1: Upper ROM region is no more accessible until next system reset
The upper ROM region is accessible only in OTP-SECURED open and OTP-SECURED closed modes and ROMLOCK bit is relevant only in these modes.

Bit 0 **OTP**: upper OTP read lock
0: upper OTP not locked
1: upper OTP cannot be reloaded until next system reset

3.6.6 BSEC debug configuration register (BSEC_DENABLE)

Address offset: 0x014

Reset value: 0x0000 0000

Note: reset value depends on OTP secure mode according to *Table 13: BSEC_DENABLE default values after reset on page 172.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DBG SWEN ABLE	CFGS DIS ABLE	CP15SDISABLE [1:0]		SPNID EN	SPID EN	HDPEN	DE VICE EN	NIDEN	DBG EN	Res.
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DBGSWENABLE**: control self hosted debug enable with signal **dbgswenable**
0: Software access to all debug components is disabled
1: Software access to all debug components is enabled

Bit 9 **CFGSDISABLE**: write access to secure GIC registers disable with signal **cfgsdisable**
0: no effect, all GIC registers are accessible
1: Disable write access to some secure GIC registers

Bits 8:7 **CP15SDISABLE[1:0]**: write access to some secure Cortex®-A7 CP15 registers disable
CPDISABLE[0] applies to CPU0. CPDISABLE[1] applies to CPU1.
0: All CP15 registers are accessible
1: Disable write access to some secure CP15 registers into Cortex®-A7 corresponding CPU

Bit 6 **SPNIDEN**: secure privilege non-invasive debug enable with signal **spiden**
0: secure privilege non-invasive debug disabled
1: secure privilege non-invasive debug enabled

Bit 5 **SPIDEN**: secure privilege invasive debug enable with signal **spniden**
0: secure privilege invasive debug disabled
1: secure privilege invasive debug enabled

Bit 4 **HDPEN**: hardware debug port enable with signal **hdpen**
0: hardware debug port disabled
1: hardware debug port enabled

Bit 3 **DEVICEEN**: controls access to debug component via external debug port by signal **deviceen**
0: disabled
1: enabled

Bit 2 **NIDEN**: non-invasive debug enable with signal **niden**
0: non-invasive debug disabled
1: non-invasive debug enabled

Bit 1 **DBGEN**: debug enable with signal **dbgen**
0: disabled
1: enabled

Bit 0 Reserved, must be kept at reset value.

3.6.7 BSEC OTP disturbed status register x (BSEC_OTP_DISTURBEDx)

BSEC_OTP_DISTURBED**0** is used to report disturbed state of BSEC_OTP_DATA0 to BSEC_OTP_DATA31 (lower 1 Kbits OTP).

BSEC_OTP_DISTURBED**1** is used to report disturbed state of BSEC_OTP_DATA32 to BSEC_OTP_DATA63.

BSEC_OTP_DISTURBED**2** is used to report disturbed state of BSEC_OTP_DATA64 to BSEC_OTP_DATA95.

Address offset: 0x01C + 0x004 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DIS[31:0]**: disturbed status of the corresponding OTP word
0: OTP word has been read correctly
1: OTP word is disturbed, either because of a read with disturbcheck failed or because the spare ECC bits [37:32] for words 0 to 31 do not match expected value. It may result from a possible hardware attack.

Note: ECC bits resilience against hardware attacks is applied only to lower OTP region.

3.6.8 BSEC OTP error status register x (BSEC_OTP_ERRORx)

BSEC_OTP_ERROR**0** is used to report error state of BSEC_OTP_DATA0 to BSEC_OTP_DATA31 (lower 1 Kbits OTP which are protected by 2:1 redundancy).

BSEC_OTP_ERROR**1** is used to report error state of BSEC_OTP_DATA32 to BSEC_OTP_DATA63 which are protected by 6-bit ECC.

BSEC_OTP_ERROR**2** is used to report error state of BSEC_OTP_DATA64 to BSEC_OTP_DATA95 which are protected by 6-bit ECC.

Address offset: 0x034 + 0x004 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ERR[31:0]**: error status of the correspond OTP word
0: OTP word has been read correctly
1: OTP word is erroneous, either because 2:1 redundancy has failed for words 0 to 31 or because ECC check has failed for words 32 to 95.

3.6.9 BSEC OTP lock status register x (BSEC_OTP_WRLOCKEDx)

BSEC_OTP_WRLOCKED**0** is used to report permanent write lock of BSEC_OTP_DATA0 to BSEC_OTP_DATA31.

BSEC_OTP_WRLOCKED**1** is used to report permanent write lock of BSEC_OTP_DATA32 to BSEC_OTP_DATA63.

BSEC_OTP_WRLOCKED**2** is used to report permanent write lock of BSEC_OTP_DATA64 to BSEC_OTP_DATA95.

Permanent write lock requires a programming sequence to lock a word (see section:*Section 3.3.6: OTP operations on page 170*).

Address offset: 0x04C + 0x004 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRLOCK[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRLOCK[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRLOCK[31:0]**: permanent word lock status of the correspond OTP word
0: OTP word not locked
1: OTP word permanently locked

3.6.10 BSEC OTP sticky programming lock register x (BSEC_OTP_SPLOCKx)

BSEC_OTP_SPLOCK**0** is used to lock the programming of BSEC_OTP_DATA0 to BSEC_OTP_DATA31 until next system-reset

BSEC_OTP_SPLOCK**1** is used to lock the programming of BSEC_OTP_DATA32 to BSEC_OTP_DATA63 until next system-reset

BSEC_OTP_SPLOCK**2** is used to lock the programming of BSEC_OTP_DATA64 to BSEC_OTP_DATA95 until next system-reset

Attempt to sticky program locked OTP word are silently ignored.

Address offset: 0x064 + 0x004 * x, (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLOCK[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLOCK[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SPLOCK[31:0]**: lock programming for the OTP word until next power-on reset
0: OTP word not locked
1: OTP word locked for programming

3.6.11 BSEC OTP shadow write sticky lock register x (BSEC_OTP_SWLOCKx)

BSEC_OTP_SWLOCK**0** is used to prevent writing to BSEC_OTP_DATA0 to BSEC_OTP_DATA31 until next system-reset.

BSEC_OTP_SWLOCK**1** is used to prevent writing to BSEC_OTP_DATA32 to BSEC_OTP_DATA63 until next system-reset.

BSEC_OTP_SWLOCK**2** is used to prevent writing to BSEC_OTP_DATA64 to BSEC_OTP_DATA95 until next system-reset.

Write to shadow write locked BSEC_OTP_DATA word are silently ignored.

Note: Writing to OTP word 0 shadow is always prevented.

Address offset: 0x07C + 0x004 * x, (x = 0 to 2)

Reset value: 0x0000 0001, 0x0000 0000, 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWLOCK[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWLOCK[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SWLOCK[31:0]**: lock the writing for the OTP shadow word until next power-on reset.

0: shadow word not locked

1: shadow word locked and cannot be written by software

Note: BSEC_OTP_SWLOCK0 bit 0 is forced to 1 by hardware, writing to this bit has no effect.

3.6.12 BSEC OTP shadow read sticky lock register x (BSEC_OTP_SRLOCKx)

BSEC_OTP_SRLOCK**0** is used to prevent reloading of BSEC_OTP_DATA0 to BSEC_OTP_DATA31 until next system-reset.

BSEC_OTP_SRLOCK**1** is used to prevent reloading of BSEC_OTP_DATA32 to BSEC_OTP_DATA63 until next system-reset.

BSEC_OTP_SRLOCK**2** is used to prevent reloading of BSEC_OTP_DATA64 to BSEC_OTP_DATA95 until next system-reset.

Setting SRLOCK bits or attempt to reload a locked OTP do **not** clear the corresponding BSEC_OTP_DATAx shadow register.

Note: BSEC_OTP_SRLOCK0 bit 0 is controlled by hardware according to fuse_ok, writing to this bit has no effect.

Address offset: 0x094 + 0x004 * x, (x = 0 to 2)

Reset value: 0x0000 000X, 0x0000 0000, 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRLOCK[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRLOCK[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SRLOCK[31:0]**: prevent reloading of the shadow word from OTP until next power-on reset.
0: OTP word not locked
1: OTP word locked and cannot be reloaded
Note: Reloading of OTP word 0 is prevented after its initial read on a system reset.

3.6.13 BSEC JTAG input register (BSEC_JTAGIN)

Address offset: 0x0AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]**: JTAG input data
16-bit copy from JTAG SoC register

3.6.14 BSEC JTAG output register (BSEC_JTAGOUT)

Address offset: 0x0B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA[15:0]**: JTAG output data
16-bit copy to JTAG SoC register

3.6.15 BSEC scratch register (BSEC_SCRATCH)

Address offset: 0x0B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: scratch data
This register is a general purpose register.

3.6.16 BSEC shadow register x (BSEC_OTP_DATAx)

The complete description of BSEC_OTP_DATAx register values and properties can be found in *Section 4: OTP mapping (OTP)*.

Several OTP directly impact BSEC behavior, such as:

- BSEC_OTP_DATA0[6:0] (see *Table 11: OTP modes definition on page 168*)

Address offset: 0x200 + 0x004 * x, (x = 0 to 95)

Reset value: 0xXXXX XXXX

The reset value depends on the actual OTP programmed value and the OTP mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: shadow register read from OTP or written by software (OTP emulation mode)
The update of shadow registers is controlled by other registers and by OTP security mode.

3.6.17 BSEC hardware configuration register (BSEC_HWCFGR)

Address offset: 0xFF0

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECC_USE[3:0]				SIZE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **ECC_USE[3:0]**: protection / redundancy scheme used
0x0: No
0x1: ECC for upper OTP bits used
Others: reserved

Bits 3:0 **SIZE[3:0]**: OTP block size
0x2: 2 Kbits
0x4: 4 Kbits
0x8: 8 Kbits
Others: reserved

3.6.18 BSEC version register (BSEC_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: major revision information

Bits 3:0 **MINREV[3:0]**: minor revision information

3.6.19 BSEC identification register (BSEC_IPIDR)

Address offset: 0xFF8

Reset value: 0x0010 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: BSEC identification

3.6.20 BSEC size identification register (BSEC_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: BSEC size identification

3.6.21 BSEC register map

Table 16 describes BSEC register map and reset values.

Table 16. BSEC register map and reset values

Offset	Register name and reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	BSEC_OTP_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREAD		PRGWDTH				FRC		PWRUP
	Reset value																								0	0	0	0	0	1	1	1	0
0x004	BSEC_OTP_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK	PROG	Res.	ADDR[6:0]						
	Reset value																							0	0		0	0	0	0	0	0	0
0x008	BSEC_OTP_WRDATA	WRDATA[31:0]																															
	Reset value																																
0x00C	BSEC_OTP_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWRON	PROGFAIL	BUSY	INVALID	Res.	SECURE
	Reset value																											0	0	0	0		0
0x010	BSEC_OTP_LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPLOCK	Res.	DENREG	ROMLOCK	OTP
	Reset value																												0		0	0	0
0x014	BSEC_OTP_DENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFGSWENABLE	CFGSDISABLE	CP15SDISABLE[1:0]		SPNIDEN	SPIDEN	HDPEN	DEVICEEN	NIDEN	\DBGEN	Res.
	Reset value																						0	0	0		0	0	0	0	0	0	
0x01C	BSEC_OTP_DISTRUBED0	DIS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	BSEC_OTP_DISTURBED1	DIS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	BSEC_OTP_DISTURBED2	DIS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	BSEC_OTP_ERROR0	ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16. BSEC register map and reset values (continued)

Offset	Register name and reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	BSEC_OTP_ ERROR1	ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	BSEC_OTP_ ERROR2	ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	BSEC_OTP_ WRLOCKED0	WRLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	BSEC_OTP_ WRLOCKED1	WRLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	BSEC_OTP_ WRLOCKED2	WRLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	BSEC_OTP_ SPLOCK0	SPLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	BSEC_OTP_ SPLOCK1	SPLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	BSEC_OTP_ SPLOCK2	SPLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	BSEC_OTP_ SWLOCK0	SWLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x080	BSEC_OTP_ SWLOCK1	SWLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	BSEC_OTP_ SWLOCK2	SWLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	BSEC_OTP_ SRLOCK0	SRLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X
0x098	BSEC_OTP_ SRLOCK1	SRLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 16. BSEC register map and reset values (continued)

Offset	Register name and reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x09C	**BSEC_OTP_ SRLOCK2**	SRLOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	**BSEC_JTAGIN**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0	**BSEC_JTAGOUT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	**BSEC_SCRATCH**	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x200	**BSEC_OTP_DATA0**	DATA[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x204	**BSEC_OTP_DATA1**	DATA[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . .	. . .	. . .																															
0x37C	**BSEC_ OTP_DATA95**	DATA[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFF0	**BSEC_IPHW_ CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECC_USE [3:0]				SIZE[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFF4	**BSEC_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFF8	**BSEC_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
0xFFC	**BSEC_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

4 OTP mapping (OTP)

The OTP bits are read or programmed by the Boot and Security and OTP control (BSEC). Before being programmed the OTP bit default is ‘0’.

The STM32MP153 device has 3072 OTP (One Time Programmable) bits which can be read-accessed in 96 words: BSEC_OTP_DATAx (x = 0 to 95):

– Lower OTP are listed in *Table 17: OTP list words 0 to 31*.

– Upper OTP are listed in *Table 18: OTP list words 32 to 95*.

The tables list the OTP word and bit field relevant properties using the following columns:

Name: OTP word synonym (for example OTP word0 is aka CFG0).

Description: OTP word or bit field description.

Prog: programming according to:

s: programmed by STMicroelectronics at factory whether product differentiations or keys.

u: available to any other software executed after bootROM.

r: programmed by bootrom (for example, during SSP process)

wplock:

n: not locked.

y: permanent programming lock by factory programming.

swprlockx: sticky lock set by boot listed as s[w][p][r][c] with character w, p, r, c according to:

w: indicates write of shadow is sticky locked,
p: indicates OTP programming is sticky locked,
r: indicates OTP reloading is sticky locked,

c: indicates OTP shadow is cleared during boot.

swprlock1: sticky lock set after boot in secured-open.

swprlock2: sticky lock set after boot in secured-closed (only possible for device with secure boot capabilities).

swprlock3: sticky lock set after boot for RMA.

Table 17. OTP list words 0 to 31

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
0	CFG0	**OTP mode encoding**							
		0 to 5	OTP mode encoding	s	n	sw	sw	sw	-
		6	Closed_device bit[1]	u					
		7 to 31	Reserved for STMicroelectronics	s					
1	CFG1	**RPN coding**							
		0 to 7	RPN coding	s	y	sw	sw	sw	*Section 65.2: Device part number (RPN)*
		8 to 31	Reserved for STMicroelectronics						
2	CFG2	**Rerserved for STMicroelectronics**							
		0 to 3	SSP_ATTEMPTS	r	y	swprc	swprc	swprc	Set by bootROM during SSP process.
		4 to 31	Reserved for STMicroelectronics	s					-
3	CFG3	**Boot / source definition**							
		0	qspi_not_default_af	u	n	-	sw	-	0: QSPI uses default hard coded AFmux 1: QSPI uses AFmux defined in OTP
		1 to 2	emmc_if_id						0: Source is default one: SDMMC2 with default AFMux 1: SDMMC1. 2: SDMMC2 (uses non default AFmux defined in OTP)
		3 to 4	sd_if_id						0: Source is default one: SDMMC1 with default AFMux 1: SDMMC1 (uses non default AFmux defined in OTP) 2: SDMMC2
		5	no_cpu_pll						0: PLLs for CPU/AXI are enable for cold boot 1: PLLs for CPU/AXI are not enable for cold boot
		6	no_usb_dp_pullup						0: USB DP pull-up is set 1: USB DP pull-up is not set
		7 to 14	uart_instances_disabled						0b00000001 reserved. 0b00000010 disable USART2. 0b00000100 disable USART3. 0b00001000 disable UART4. 0b00010000 disable UART5. 0b00100000 disable UART6. 0b01000000 disable UART7. 0b10000000 disable USART8.
		15	no_data_cache						0: Data cache is used by bootrom. 1: Data cache is not used by bootrom.

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
		Boot / source definition (continued)							
3	CFG3	16 to 23	boot_source_disable	u	n	-	sw	-	0b00000001 disable FMC boot source. 0b00000010 disable QSPI NOR boot source. 0b00000100 disable eMMC boot source. 0b00001000 disable SD boot source. 0b00010000 disable UART boot source. 0b00100000 disable USB boot source. 0b01000000 disable QSPI NAND boot source.
		24 to 26	secondary_boot_source						0: No secondary boot source is defined 1: FMC NAND 2: QSPI NOR 3: eMMC 4: SD 5: QSPI NAND
		27 to 29	primary_boot_source						0: No primary boot source is defined 1: FMC NAND 2: QSPI NOR 3: eMMC 4: SD 5: QSPI NAND
		30 to 31	HSE value						0b00 HSE is autodetected. 0b01 HSE is 24 MHz. 0b10 HSE is 25 MHz. 0b11 HSE is 26 MHz.
4	CFG4	**Boot / monotonic counter**							
		0 to 31	Boot / monotonic counter	u	n	-	-	-	Value of monotonic counter is 31 is 2*X where X is position of the most significant bit at 1.

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
		Boot / AFmux configuration 1							
5	CFG5	0 to 3	mode0	u	n	-	sw	-	0: AF; no pull; low speed 1: AF; no pull; medium speed 2: AF; no pull; high speed 3: AF; pull up; low speed 4: AF; pull up; medium speed 5: AF; pull up; high speed 6: AF; pull down; low speed 7: AF; pull down; medium speed 8: AF; pull down; high speed 9: GPIO output high 10: GPIO output low 11: GPIO input 12: GPIO open drain; No pull 13: GPIO open drain; pull up 14: GPIO open drain; pull down 15: GPIO analog mode
		4 to 7	afmux0						AF mux value between 0 and 15.
		8 to 11	pin0						pin id between 0 and 15 for GPIOA to GPIOJ and between 0 and 7 for GPIOK and GPIOZ.
		12 to 15	port0						0: unused (in that case, mode0, afmux0 and pin0 fields could be used as general purpose OTP) 1: Bank A 2: Bank B 3: Bank C 4: Bank D 5: Bank E 6: Bank F 7: Bank G 8: Bank H 9: Bank I 10: Bank J 11: Bank K 12: Bank Z Others: Invalid configuration
		16 to 19	mode1						idem CFG5.mode0
		20 to 23	afmux1						idem CFG5.afmux0
		24 to 27	pin1						idem CFG5.pin0
		28 to 31	port1						idem CFG5.port0

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
6	CFG6	**Boot / AFmux configuration 2**							
6	CFG6	0 to 3	mode2	u	n	-	sw	-	idem CFG5.mode0
6	CFG6	4 to 7	afmux2	u	n	-	sw	-	idem CFG5.afmux0
6	CFG6	8 to 11	pin2	u	n	-	sw	-	idem CFG5.pin0
6	CFG6	12 to 15	port2	u	n	-	sw	-	idem CFG5.port0
6	CFG6	16 to 19	mode3	u	n	-	sw	-	idem CFG5.mode0
6	CFG6	20 to 23	afmux3	u	n	-	sw	-	idem CFG5.afmux0
6	CFG6	24 to 27	pin3	u	n	-	sw	-	idem CFG5.pin0
6	CFG6	28 to 31	port3	u	n	-	sw	-	idem CFG5.port0
7	CFG7	**Boot / AFmux configuration 3**							
7	CFG7	0 to 3	mode4	u	n	-	sw	-	idem CFG5.mode0
7	CFG7	4 to 7	afmux4	u	n	-	sw	-	idem CFG5.afmux0
7	CFG7	8 to 11	pin4	u	n	-	sw	-	idem CFG5.pin0
7	CFG7	12 to 15	port4	u	n	-	sw	-	idem CFG5.port0
7	CFG7	16 to 19	mode5	u	n	-	sw	-	idem CFG5.mode0
7	CFG7	20 to 23	afmux5	u	n	-	sw	-	idem CFG5.afmux0
7	CFG7	24 to 27	pin5	u	n	-	sw	-	idem CFG5.pin0
7	CFG7	28 to 31	port5	u	n	-	sw	-	idem CFG5.port0
8	CFG8	**Partially reserved for STMicroelectronincs**							
8	CFG8	0 to 7	Reserved for STMicroelectronics	s	n	swp	swp	swp	-
8	CFG8	8	SSP_REQ	u	n	swp	swp	swp	Set by FSBL(TF-A) during SSP process.
8	CFG8	9	SSP_SUCCESS	u	n	swp	swp	swp	Set by FSBL(TF-A) during SSP process.
8	CFG8	10 to 31	Reserved for STMicroelectronics	s	n	swp	swp	swp	-

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
9	CFG9	**BootROM / configuration**							
		0	Boot_traces_disabled	u	n	-	-	-	0: no. 1: yes
		1	hse_frequency_auto detection_disable	u					0: no. 1: yes
		2	hse_bypass_detection_disable	u					0: no. 1: yes
		3	Reserved for STMicroelectronics	s					-
		4 to 13	not used yet	s					-
		14	spinand_need_plane_select	u					0: SPI NAND plane select not need 1: SPI NAND plane select is needed
		15 to 17	nand_number_of_ecc_bits	u					0: ECC unset. 1: ECC 1bit (Hamming) 2: ECC 4bit (BCH4) 3: ECC 8bit (BCH8) 4: on-die ECC
		18	nand_bus_width	u					0: 8 bit 1: 16 bit
		19 to 26	nand_nb_of_blocks	u					Number of blocks in unit of 256 blocks.
		27 to 28	nand_block_size	u					Block size in number of pages: 0: 64 pages per block 1: 128 pages per block 2: 256 pages per block
		29 to 30	nand_page_size	u					0: 2Kbytes 1: 4Kbytes 2: 8Kbytes
		31	nand_param_stored_in_otp	u					0: no 1: nand parameters bits 15 to 30 are used
10	CFG10	**Device configuration**							
		0 to 31	for OEM device configuration	u	n	-	-	-	-
11	CFG11	**Device configuration**							
		0 to 31	for OEM device configuration	u	n	-	-	-	-
12	CFG12	**Engineering purposes**							
		0 to 31	Reserved for STMicroelectronics	s	y	-	-	-	-
13	ID0	**Unique device ID (96-bit)**							
		0 to 31	UID[31:0]	s	y	-	-	-	-
14	ID1	**Unique device ID (96-bit)**							
		0 to 31	UID[63:32]	s	y	-	-	-	-

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
15	ID2	**Unique device ID (96-bit)**							
		0 to 31	UID[95:64]	s	y	-	-	-	-
16	HW0	**Hardware trimming / analog 1**							
		0 to 5	VREFBUF trimming	s	y	swp	swp	swp	VREBUF Trim
		6 to 9	I/O compensation trimming						Used for I/O calibration
		10 to 11	Reserved						-
		12 to 15	1V8 reference trimming						1V8 Regulator Trim
		16 to 21	LSI trimming						-
		22 to 26	1V1 reference trimming						1V1 Regulator Trim
		27 to 29	PKG: factory programmed						Refer to *Section 65.4: Package data register (PKG)*.
		30 to 31	Reserved for STMicroelectronics						-
17	HW1	**Hardware trimming / analog 2**							
		0 to 7	CSITRIM[7:0]	s	y	-	-	-	CSI Trim
		8 to 19	HSITRIM[11:0]						HSI Trim
		20 to 24	DAC1 factory trimming						DAC1 Trim
		25 to 29	DAC2 factory trimming						DAC2 Trim
		30 to 31	Reserved for STMicroelectronics						-

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
18	HW2	**Hardware trimming / PKG / WDG config**							
		0 to 2	Reserved for STMicroelectronics	s	n	-	-	-	-
		3	IWDG1_HW	u					IWDG1 start: 0: start by software 1: auto start
		4	IWDG2_HW						IWDG2 start: 0: start by software 1: auto start
		5	IWDG1_FZ_STOP						IWDG1 freeze in Stop: 0: no 1: freeze in Stop
		6	IWDG2_FZ_STOP						IWDG2 freeze in Stop: 0: no 1: freeze in Stop
		7	IWDG1_FZ_STANDBY						IWDG1 freeze in Standby: 0: no 1: freeze in Standby
		8	IWDG2_FZ_STANDBY						IWDG2 freeze in Standby: 0: no 1: freeze in Standby
		9	OTP_MCU_SYSRST_EN						MCU allowed to generate System Reset.
		10	Reserved for STMicroelectronics	s					-
		11 to 12	SELINBORH[1:0]	u					00: BOR Disabled 01: BOR = 2.1 V 10: BOR = 2.4 V 11: BOR = 2.7 V
		13	PRODUCT_BELOW_2V5						1: required when VDD is below 2.5 V to allow the SYSCFG IO control register HSLVEN_xxx bits to be taken into account. For more information see *Section 14.3: SYSCFG registers* SYSCFG_IOCTRLSETR and SYSCFG_IOCTRLCLRR registers.
		14 to 31	Reserved for STMicroelectronics	s					-
19	HW3	**Hardware trimming / calibration for DTS**							
		0 to 15	TS1_FMT0[15:0]	s	y	-	-	-	DTS frequency measured freq at T0
		16 to 31	TS1_RAMP_COEF[15:0]	s	y	-	-	-	DTS ramp coefficient

Table 17. OTP list words 0 to 31 (continued)

Word	Name	Bits	Description	prog	wp lock	swpr lock1	swpr lock2	swpr lock3	Detail description
20	HW4	**Hardware trimming / calibration for V_{REFINT}, DTS**							
		0 to 6	TRIM_DTP_R[6:0]	s	y	-	-	-	DTS R Trim
		7 to 8	TS1_T0[1:0]	s	y	-	-	-	DTS T0 value
		9 to 15	Reserved for STMicroelectronics	s	y	-	-	-	-
		16 to 31	VREFINT_CAL[15:0]	s	y	-	-	-	ADC VBG measurement to correct for VREF
21	HW5	**Memory repair word 1**							
		31 to 0	Reserved for STMicroelectronics	s	y	-	-	-	-
22	HW6	**Memory repair word 2**							
		0 to 31	Reserved for STMicroelectronics	s	y	-	-	-	-
23	HW7	**Hardware trimming / calibration for analog temp sensor**							
		0 to 15	TS_CAL1[15:0]	s	y	-	-	-	ADC TEMP measurement at T0
		16 to 31	TS_CAL215:0]	s	y	-	-	-	ADC TEMP measurement at T1
24	PKH0	-							
		0 to 31	Public Key Hash(2)	u/s	n/y(3)	sw	sw	sw	-
25	PKH1	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-
26	PKH2	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-
27	PKH3	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-
28	PKH4	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-
29	PKH5	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-
30	PKH6	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-
31	PKH7	-							
		0 to 31	Public Key Hash	u/s	n/y(3)	sw	sw	sw	-

1. This bit must never be programmed to 1 on products without secure-boot option available (refer to product datasheet). As an example, setting this bit on STM32MP15xA and STM32MP15xD products might make then permanently fail to boot. To avoid this case, it is recommended to make the OTP Word 0 permanent write locked (see *Section : OTP permanent write lock* in BSEC chapter).
2. The Public Key Hash is composed of 32 bytes: hash = byte0 byte1 byte2 ... byte31. These bytes are stored in the OTP in 8 words of 4 bytes each. The first four bytes of the hash table are stored in PKH0, the next four in PKH1, and so on. Within each word the lowest numbered byte is stored in the most significant position, and the highest numbered byte in the least significant position. Example: if hash = 00 01 02 03 04 05 06 07 08... then PKH0 = 0x00010203, PKH1 = 0x04050607, and so on.
3. This word is permanent write locked on devices without secure-boot option (see datasheet).

Table 18. OTP list words 32 to 95

Word	Name	Bits	Description	prog	wplock	swpr lock1	swpr lock2	swpr lock3
32	XK0	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
33	XK1	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
34	XK2	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
35	XK3	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
36	XK4	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
37	XK5	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
38	XK6	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
39	XK7	0 to 31	Reserved for STMicroelectronics	s	y	swprc	swprc	swprc
40	XK8	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
41	XK9	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
42	XK10	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
43	XK11	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
44	XK12	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
45	XK13	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
46	XK14	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
47	XK15	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
48	XK16	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
49	XK17	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
50	XK18	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
51	XK19	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
52	XK20	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp

Table 18. OTP list words 32 to 95 (continued)

Word	Name	Bits	Description	prog	wplock	swpr lock1	swpr lock2	swpr lock3
53	XK21	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
54	XK22	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
55	XK23	**ST Public ECDSA Chip Certificate for SSP**						
		0 to 31	Certificate bits(1)	s	y	swp	swp	swp
56	XK24	**Password for RMA**						
		0 to 14	RMA unlock password	u	n	swrc	swrc	swrc
		15 to 29	RMA relock password					
		30 to 31	Not used	s				
57	XK25	**MAC address 1**						
		0 to 31	mac[31:0]	u	n	c	c	swprc
58	XK26	**MAC address 2**						
		0 to 15	mac[47:32]	u	n	c	c	swprc
		16 to 31	Free for user					
59	XK27	-	Free for user	u	n	c	c	swprc
60	XK28	-	Free for user	u	n	c	c	swprc
61	XK29	-	Free for user	u	n	c	c	swprc
62	XK30	-	Free for user	u	n	c	c	swprc
63	XK31	-	Free for user	u	n	c	c	swprc
64	XK32	-	Free for user	u	n	c	c	swprc
65	XK33	-	Free for user	u	n	c	c	swprc
66	XK34	-	Free for user	u	n	c	c	swprc
67	XK35	-	Free for user	u	n	c	c	swprc
68	XK36	-	Free for user	u	n	c	c	swprc
69	XK37	-	Free for user	u	n	c	c	swprc
70	XK38	-	Free for user	u	n	c	c	swprc
71	XK39	-	Free for user	u	n	c	c	swprc
72	XK40	-	Free for user	u	n	c	c	swprc
73	XK41	-	Free for user	u	n	c	c	swprc
74	XK42	-	Free for user	u	n	c	c	swprc
75	XK43	-	Free for user	u	n	c	c	swprc
76	XK44	-	Free for user	u	n	c	c	swprc
77	XK45	-	Free for user	u	n	c	c	swprc

Table 18. OTP list words 32 to 95 (continued)

Word	Name	Bits	Description	prog	wplock	swpr lock1	swpr lock2	swpr lock3
78	XK46	-	Free for user	u	n	c	c	swprc
79	XK47	-	Free for user	u	n	c	c	swprc
80	XK48	-	Free for user	u	n	c	c	swprc
81	XK49	-	Free for user	u	n	c	c	swprc
82	XK50	-	Free for user	u	n	c	c	swprc
83	XK51	-	Free for user	u	n	c	c	swprc
84	XK52	-	Free for user	u	n	c	c	swprc
85	XK53	-	Free for user	u	n	c	c	swprc
86	XK54	-	Free for user	u	n	c	c	swprc
87	XK55	-	Free for user	u	n	c	c	swprc
88	XK56	-	Free for user	u	n	c	c	swprc
89	XK57	-	Free for user	u	n	c	c	swprc
90	XK58	-	Free for user	u	n	c	c	swprc
91	XK59	-	Free for user	u	n	c	c	swprc
92	XK60	-	Free for user	u	n	c	c	swprc
93	XK61	-	Free for user	u	n	c	c	swprc
94	XK62	-	Free for user	u	n	c	c	swprc
95	XK63	-	Free for user	u	n	c	c	swprc

1. The ST Public ECDSA Chip Certificate for SSP is composed of 64 bytes: key = byte0 byte1 byte2 ... byte63. These bytes are stored in the OTP in 16 words of 4 bytes each. The first four bytes of the key table are stored in XK8, the next four in XK9, and so on. Within each word the lowest numbered byte is stored in the most significant position, and the highest numbered byte in the least significant position. Example: if key = 00 01 02 03 04 05 06 07 08 ... then XK8 = 0x00010203, XK9 = 0x04050607, and so on.

5 DDR3/LPDDR2/LPDDR3 controller (DDRCTRL)

5.1 Introduction

The DDRCTRL is a DDR-SDRAM controller and specific configuration of Synopsys uMCTL2 DDR controller v3.00a.

The DDRCTRL combined with the DDRPHYC (DDR PHY with Physical Utility Block) provides a complete memory interface solution for the DDR memory subsystem.

5.2 DDRCTRL features

- Two 64-bit AMBA® 4 AXI4 ports interface (XPI)
- AXI clock asynchronous to the controller
- Supported standards:
  - JEDEC DDR3 SDRAM specification, JESD79-3E for DDR3/3L with 32-bit interface
  - JEDEC LPDDR2 SDRAM specification, JESD209-2E for LPDDR2 with 32-bit interface
  - JEDEC LPDDR3 SDRAM specification, JESD209-3B for LPDDR3 with 32-bit interface

Warning: The STM32MP153x does not support the ODT for LPDDR3

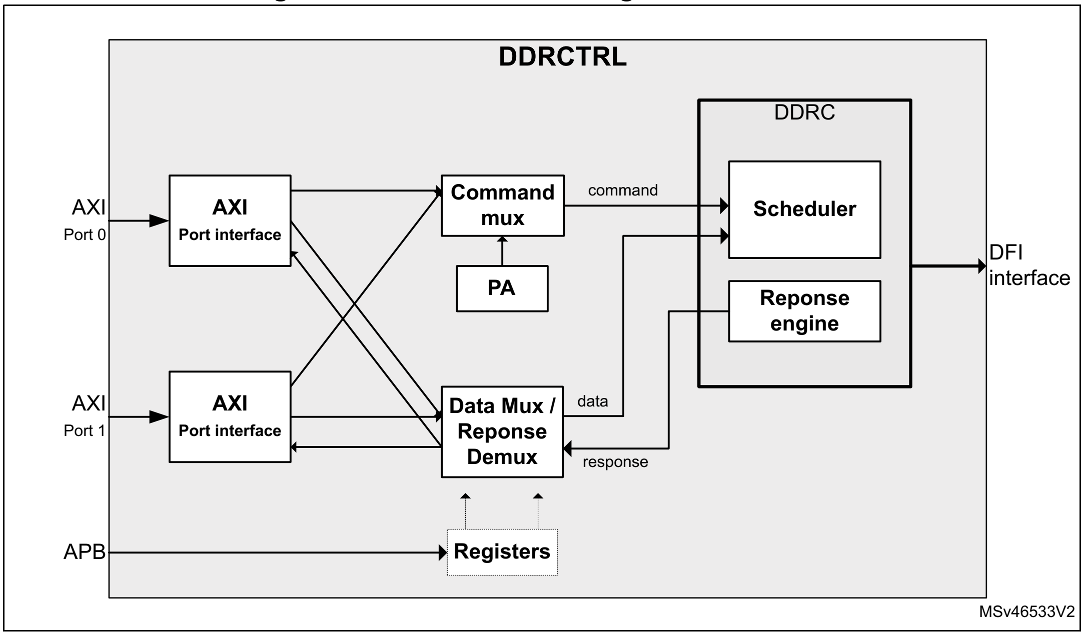
- 64-bit with DFI 2.1 compliant interface to DDRPHYC
- Advanced scheduler and SDRAM command generator
- 1:1 frequency ratio between the DDRC clock and the DDR PHY / external memory clock (aka SDR mode)
- Asynchronous clock between the AXI ports and the DDRC (the clock crossing is local to XPI's)
- Programmable full data width (32-bit) or half data width (16-bit)
- Advanced QoS support with 3 traffic class on read and 2 traffic classes on write
- Options to avoid starvation of lower priority traffic
- Guaranteed coherency for write-after-read (WAR) and read-after-write (RAW) on the AXI ports
- Programmable support for burst length options (4, 8,16)
- Write combine to allow multiple writes to the same address to be combined into a single write
- Single rank configuration
- Supports automatic SDRAM power-down entry and exit caused by a lack of transaction arrival for programmable time
- Supports automatic clock stop (LPDDR2/3) entry and exit caused by lack of transaction arrival

- Supports automatic low-power mode operation caused by a lack of transaction arrival for programmable time via hardware low-power interface
- Programmable paging policy selectable as any of the following:
  - Leave page open after accesses (Open page policy)
  - Close page when there are no further accesses available in the controller for that page
  - Auto-precharge with each access, with an optimization for page-close mode which leaves the page open after a flush for read-write and write-read collision cases
- Supports the self-refresh entry and exit as follows:
  - Automatic self-refresh entry and exit caused by lack of transaction arrival for programmable time
  - Self-refresh entry and exit under software control
- Support for deep power-down entry and exit under software control (LPDDR2)
- Support for explicit SDRAM mode register updates under software control
- Flexible address mapper logic to allow application specific mapping of row, column, bank bits
- Programmable support for 1T or 2T timing
- User-selectable refresh control options:
  - Controller-generated auto-refreshes at programmable average intervals
  - Ability to group up to 8 controller-generated refreshes together to be issued consecutively (this reduces the frequency of page closings, increasing the overall efficiency)
  - When the controller-generated refreshes are grouped, some refreshes can be issued speculatively when the controller is idle for a programmable period of time
  - Ability to disable controller-generated auto-refreshes
  - Ability to issue a refresh through direct software request
  - User-selectable ability to perform per-bank refreshes rather than all-banks refreshes for LPDDR2/3

5.3 DDRCTRL block diagram

A simplified DDRCTRL block diagram is shown in *Figure 7*.

Figure 7. DDRCTRL block diagram



5.4 DDRCTRL architecture overview

See the DDRCTRL block diagram in *Figure 7*.

- The AXI Port Interface (XPI) block provides the interface to the AXI port with the AXI bus protocol handling, data buffering and reordering for read data, data bus size / clock domain conversion and memory burst address alignment.
- The Port Arbiter (PA) block provides latency sensitive, priority based arbitration between the addresses issued by the XPI's.
- The DDR Controller (DDRC) block contains a logical CAM (Content Addressable Memory). This holds information on the commands, which is used by the scheduling algorithms to optimally schedule commands to be sent to the PHY, based on priority, bank/rank status and DDR timing constraints.
- The APB register block contains the software accessible registers
- The write data are stored until its associated command is issued to the PHY. The Write data buffer is local to DDRC.
- The read data are handled by the response engine in the DDRC and returned in the order of scheduled read commands on the HIF. the read data are stored in the read re-order buffer (local to XPI) and returned in order, to the AXI ports.

5.4.1 AXI Port Interface (XPI)

The XPI processes the AXI information: address, burst size, burst length and burst type and converts the AXI burst into read and write requests which are forwarded to the PA.

In the opposite direction, the XPI converts the responses from the DDRC into appropriate AXI responses.

Read address channel features:

- Supports any burst length for incremental bursts (<=256 byte) and burst with length <= 16 for WRAP bursts.
- The start address can be unaligned to the AXI width.
- The burst size can be less than the AXI width.
- The Read Address Queue (RAQ) is configured with dual queues (named “red” and “blue” queues)
- The clock crossing is performed in RAQ
- The depth of RAQ is set to 4.
- The generation of read request is based on AXI address and burst information and divided into packets equal to memory burst length (BL2, BL4, BL8, BL16)
- In case of an unaligned burst, the first read request is unaligned and the remaining read requests are aligned.

  In general, the re-alignment to a memory burst boundary potentially causes some data beats to be discarded (affecting bandwidth) and potentially introduces additional latency on the read data and response channel.

Write address channel features:

- The Write Address Queue (WAQ) is used to store all the addresses for write requests from a given port.
- There is a single queue for all AXI IDs from a given port.
- The write address channel behavior is similar to the read address channel.
- The clock crossing is performed in WAQ
- The depth of WAQ is set to 4.

Read data and response channels:

- The read data and response channel has a single data storage queue (RDQ)
- The clock crossing is performed in RDQ
- The depth of RDQ is set to 10
- The data from different IDs are stored in the same queue and are returned in the order of read address acceptance.
- A read reorder buffer is implemented in each port to reorder the read data for that port to the same order as the order of the AXI read commands
- The virtual channels and dynamic mapping (allow optimum reordering of read data)

5.4.2 Port arbiter (PA)

The PA arbitrates command requests from the 2 AXI ports to the HIF of the DDR Controller.

The PA is comprised of multiple tiers of arbitration stages which include:

a) Read/write arbitration (single HIF option)

b) 2 priority level arbitration based on port aging and expired-VPR/expired-VPW commands (timeout - priority0)

c) 2 priority level arbitration for read requests based on DDRC read priorities (HPR/LPR-VPR)

d) 32-priority level arbitration based on internal port aging or 16-priority level arbitration based AXI QoS inputs (programmed in AXI interconnect)

e) Round-robin arbitration to resolve ports having the same priority after passing all stages of arbitration.

Note: The port aging refers to counting down the time when a port has a request, but is not granted access to the DDRC.

The port timeout refers to when the port age becomes 0, which is also referred as the highest priority -priority0'.

The expired-VPR refers to Variable Priority Read commands whose counter has expired. The read/write credits refer to the DDRC 'Credit Mechanism' employed at the HIF interface.

5.4.3 Host Interface (HIF)

The HIF is an internal interface to the DDRCTRL (the AXI Port Interface and Port Arbiter handle all signals on this interface). It is only described for the sake of understanding QoS and multi-tier arbitration.

The read, write requests and write data are transferred from PA to DDRC using HIF interface.
The requests are throttled by:

a) A credit mechanism that ensures that the buffer space is available for any request made to the DDRC, prior to the request being made.

b) An independent stall mechanism that throttles requests when address collisions take place or when entering a self-refresh via the software self-refresh.

The read data are sent back via a common response interface

Credit mechanism:

The DDRC employs a credit mechanism to ensure that buffers do not overflow. The interface making the request to the DDRC, can only request commands for which it has been granted credits to issue.

The credits are tracked separately for the following three command types:

- High priority read (HPR)
- Low priority read (LPR)
- Write (WR)

The interface logic includes counters to track the number of credits. The credit count increments every time that the DDRC issues a credit.

The VPR commands are counted in the LPR credit bucket, the VPW commands are counted in the WR credit bucket.

Read requests:

When the LPR or HPR credit count is greater than zero, the interface can issue read requests to the DDRC

Write requests:

When the WR credit count is greater than zero, the interface can issue write requests to the DDRC.

5.4.4 DDR scheduler (DDRC)

The DDRC queues commands for scheduling and the SDRAM command generation unit. The DDRC includes 3 stores:

- HPR and LPR stores in common read CAM
- WR store in write CAM

Both CAMs are 16 entries, the HPR/LPR split is software programmable.

The DDRC allows the user to manage the Transaction Service Control (TSC) according to QoS and timeouts.

5.4.5 APB Interface (APB)

There are three classes of registers in DDRCTRL:

- Dynamic Registers
- Quasi Dynamic Registers
- Static Registers

The dynamic registers can be written at any time during operation.

The static registers can be written only when the controller is in reset.
The quasi dynamic registers can be written when the DDRCTRL is in reset or under specific conditions after the reset is done. To write the quasi dynamic registers, the DDRCTRL_SWCTL.SW_DONE bit must be set to 0 at the beginning of the programming sequence and set back to 1 at the end. After the DDRCTRL_SWSTAT.SW_DONE_ACK must be polled to acknowledge that the write action has been completed.

For more information about dynamic and quasi dynamic registers, refer to uMCTL2 Data Book [1]

5.4.6 DFI Interface (DFI)

The DDRCTRL contains a DFI MC (Memory Controller) interface, which is used to connect to a DFI compliant DDRPHYC to transfer address, control and data to the PHY.

The DFI interface is sub-divided into the following interface groups:

- Control interface
- Write data interface
- Read data interface
- Update interface
- Status interface
- Training interface (PHY-independent mode as supported by uMCTL2)
- Low-power control interface

The DDRCTRL, including its DFI interface, runs at the single data rate (SDR) clock (referred to as 1:1 frequency ratio mode in the DFI Specification), so all DFI command and address signals are equal in width to the equivalent DDR SDRAM signals. The DFI data signals are twice the width of the equivalent DDR SDRAM signals.

The DDRCTRL runs with a SDR clock and the DDRPHYC operates the SDR to DDR rate conversion on the data buses. In 1:1 frequency ratio mode, core_ddrc_core_clk and CK/CK# run at the same SDR clock frequency.

5.5 DDRCTRL functional description

5.5.1 Transaction service control (TSC) and quality of service (QoS)

The transaction service control allows the user to carefully manage the following:

- Costly read/write bus turn-around
- Priorities of read requests to generally favor high priority traffic while also preventing starvation of low priority traffic

This functionality is implemented in a simple 2-state machine for each traffic type with completely configurable controls. The states of the 2-state machine determine when reads/writes are serviced and the relative priority (high priority versus low priority reads) at any given moment.

The transactions are separated into five traffic classes: with 3 read classes and 2 write classes:

- Low priority reads (LPR)
- Variable priority reads (VPR)
- High priority reads (HPR)
- Normal priority writes (NPW)
- Variable priority writes (VPW)

The traffic classes are determined according to AXI QoS values (AxQOS[3:0]) set by the SOC AXI Interconnect (AXIM) and QoS regions settings.

The HPR and VPR/LPR are stored in a common read CAM which is split according to DDRCTRL_SCHED.lpr_num.

The VPR commands move to the expired-VPR after a timeout set by DDRCTRL_PERFVPR1.vpr_timeout. The expired -VPR goes to highest priority (Priority0).

VPW and NPW are stored in a common write CAM; the VPW commands go to the expired-VPR after timeout set by DDRC_PERFVPW1.vpw_timeout. The expired-VPW commands go to highest priority (Priority0).

Following is the list of registers related to transaction service and QoS control:

- DDRCTRL_SCHED: main scheduler control
- DDRCTRL_SCHED1: page Ccose timer
- DDRCTRL_PERFHPR1: controls HPR starvation and run_length
- DDRCTRL_PERFHPR1: controls HPR starvation and run_length
- DDRCTRL_PERFLPR1: controls LPR starvation and run_length
- DDRCTRL_PCFGR_0/1: AXI port read control for port 0/1
- DDRCTRL_PCFGW_0/1: AXI port write control for port 0/1
- DDRCTRL_PCFRQOS0_0/1 and DDRCTRL_PCFRQOS1_0/1: AXI port read QoS region mapping for port 0/1 and port timeouts.
- DDRCTRL_PCFWQOS0_0/1 and DDRCTRL_PCFWQOS1_0/1: AXI port write QoS region mapping for port 0/1 and port timeouts.

5.5.2 Paging policy options

Open page policy is set by DDRC_SCHED.pageclose = 0.

Any bank remains open until there is a need to close it because of a page miss or refresh timeout.

Intelligent precharge is a mid way between open and closed page policies, set by DDRC_SCHED.pageclose = 1 and DDRC_SCHED.pageclose_timer value to give the user control over the time waited when there are no page hits to a bank in the CAM before the auto-precharge or the explicit precharge is scheduled.

This timer allows the page to be kept open for a configurable number of clock cycles after there are no commands pending in the CAM to a bank. If there are pending commands higher up in the stream (for example XPI/PA) to the same bank/page, it gives a chance to be scheduled by the DDRCTRL as an open page command.

5.5.3 Power saving features

The DDRCTRL supports various methods to save power within the system using power saving opportunities in the SDRAM, DDRPHYC and the SOC.

Precharge Power Down: (PDN)

The DDRCTRL automatically puts SDRAM in the Precharge Power Down mode under the control of the following parameters:

- DDRCTRL_PWRCTRL.powerdown_en: to enter PDN (PDE) after a number of idle cycles
- DDRCTRL_PWRTMG.powerdown_to_x32: number of idle clocks in multiple of 32 to enter PDN
- DDRCTRL_DRAMTMG1.txp: minimum time to wait after PDN exit (PDX) to any command (DLL stays locked during PDN)

Note: During PDN refresh commands are performed and clocks must be maintained; the DDRCTRL pre-charges all banks before Precharge Power Down Exit (PDE).
The DDR3 slow power down (by setting MR0[12] =1) is not supported.

Self-Refresh: (SR)

The SR is a significant power saving opportunity for SDRAM. Furthermore during SR, the DLL's in DDRPHY C can be bypassed and the DDRCTRL clocks can be stopped.

The DDRCTRL puts SDRAM in Self Refresh (SR) in the following cases:

- Automatic SR when there are no read/write for a predefined time. This mode is controlled by:
  - DDRCTRL_PWRCTRL.selfref_en: to enter SR (SRE) after a number of idle cycle
  - DDRCTRL_PWRTMG.selfref_to_x32: number of idle clocks in multiple of 32 to enter SR.
- Software controlled SR: this mode is controlled by:
  - DDRCTRL_PWRCTRL.selfref_sw: setting this bit is referred as software self_refresh entry.
- Hardware controlled SR: this mode is supported by:
  - The FSM external to the DDRCTRL and under the control of RCC (refer to *Section 10: Reset and clock control (RCC)* for mode details). When the SoC goes into Stop mode, the DDRCTRL can put the SDRAM into self-refresh.

Deep power down: (DPD)

Note: DPD is applicable only to LPDDR2/3.
This power saving opportunity may be supported but does not present any significant interest because the SDRAM content is lost during DPD, practically DPD is not different from shutting down DDR

DDR3 DLL off mode:

The DDR3 DLL off mode may be supported at low frequencies, it requires MRS (MR1[0] =1) to disable DLL (by default DLL are on). It should use the software clock removal sequence:

```
Write 0 to DDRC_PCTRLn.port_en //to block AXI port n
Poll DDRC_PSTAT.rd_port_busy = 0
Poll DDRC_PSTAT.wr_port_busy = 0
Write 1 to DDRC_PWRCTL.selref_en // move to self-refresh
Poll DDRC_PSTAT.selfref_type = 2b10
... change AXI and DDRC clocks to lower value used w/ DLL OFF
... MRS to disable DLL
Write 0 to DDRC_PWRCTL.selref_en // exit self-refresh
Poll DDRC_PSTAT.selfref_type = 2b00
Write 1 to DDRC_PCTRLn.port_en // to unlock AXI ports
```

DDRPHY DLL bypass mode:

It is possible to disable DDRPHYC DLL's to save significant power at low frequencies; this mode must be controlled by software. The software sequence transitions SDRAM to Self-refresh during which clock frequency change is possible.

5.5.4 Address mapper

The DDRCTRL address mapper is converting the AXI port read and write request into row, bank, column SDRAM address at XPI level.

The address mapping is very flexible and defined according to DDR_ADDRMAP1 to DDR_ADDRMAP11 register settings.

The usual non interleaved Row/Bank/Column (RBC) and Bank/Row/Column (BRC) can be simply programmed according to SDRAM architecture (row/bank/column) and full or half width interface mode.

5.5.5 DRAM timing parameters

The DDRCTRL schedule commands to SDRAM according to multiple timings constrains as programmed in registers DDRCTRL_DRAMTMGn according to SDRAM JEDEC timing parameters versus frequency.

The following registers are related to timings constrains

– DDRCTRL_DRAMTMG0 to DDRCTRL_DRAMTMG14
– DDRCTRL_ODTTMG
– DDRCTRL_DFITMG0 and DDRCTRL_DFITMG1

5.5.6 SDRAM initialization sequence

The initialization is controlled by the DDRPHYC supporting a PUBL built-in initialization sequence for each DDR type and triggered by PIR registers. For more information see *Section 7: DDR physical interface control (DDRPHYC)*.

5.5.7 Refresh controls

The refresh can be issued using the auto-refresh feature in DDRCTRL or using the direct software request of the refresh command. DDRCTRL_RFSHCTL3.dis_auto_refresh selects the refresh method. When this bit is set to 1, uMCTL2 uses direct software request of refresh command. When it is set to 0, the internal auto-refresh feature is used.

Refresh using the direct software request of refresh command

Follow the steps to put the DDRCTRL in direct software request of refresh command mode:

1. Set DDRCTRL_RFSHCTL3.dis_auto_refresh bit to 1. When the register bit set, DDRCTRL checks for any pending refreshes. Any pending refreshes are issued right away using the 'critical refresh' feature inside DDRCTRL. After these refreshes are issued, all the refresh timers inside DDRCTRL are reset to 0. They are re-activated only when the auto-refresh feature is enabled.
2. The SoC core must keep track of the refresh requirements of the SDRAM.
3. The refresh command can be issued by setting the register bits DDRCTRL_DBGCMD.rank0_refresh to 1. When the refresh request is stored, the corresponding register bit is automatically cleared. The SoC core can initiate a refresh operation only if DDRCTRL_DBGSTAT.rank0_refresh_busy is low. DDRCTRL issues refresh to the SDRAM at the earliest.
4. Software-driven refresh commands are loaded into a 9-entry buffer, and are issued by DDRCTRL on the DFI as soon as it is legal to do so. If the buffer saturates, the DBGSTAT.rank0_refresh_busy remains asserted to prevent software from initiating further refreshes.

Refresh using the auto refresh feature inside DDRCTRL

The DDRCTRL provides advanced refresh controls. Besides fully-configurable refresh constraints (tRFC(min) and tREFI), it can also be programmed to gather refreshes to each rank of SDRAM to reduce the bandwidth consumed by refreshes and to increase the likelihood that refreshes can be serviced during an idle period.

The following registers are related to refresh controls:

– DDRCTRL_RFSHCTL0
– DDRCTRL_RFSHCTL3
– DDRCTRL_RFSHTMG

The purposes of refresh control are to:

- reduce the bandwidth impact of refresh cycles
- increase the likelihood of refreshes being serviced during idle periods
- provide fine-gain control of the trading-off the above benefits (to gather refreshes) versus the increased worst case latencies associated with gathering refreshes

For more information, refer to the register description; besides issuing refresh at regular intervals according to tRFC(min) and tREFI, several optimization options are possible: Burst refresh, Speculative refresh, and Per-bank refresh (applicable to LPDDR2/3).

5.5.8 ZQ calibration

This feature is applicable to all DDR types. The DDRCTRL can use the ZQ calibration command to calibrate the SDRAM RON (resistor ON) and ODT (on-die termination) values over PVT (process, voltage, temperature).

The DDRPHYC supports the RON and ODT calibration at initialization; for more information see *Section 7: DDR physical interface control (DDRPHYC)*.

The DDRCTRL may issue periodic calibrations which take less time at regular interval. (ZQCS).

The DDRCTRL may also issue a long calibration (ZQCL) after self-refresh exit

The following registers are related to ZQ calibration controls:

– DDRCTRL_ZQCTL0
– DDRCTRL_ZQCTL1
– DDRCTRL_ZQCTL2
– DDRCTRL_ZQSTAT

For more information on ZQ Calibration, refer to JEDEC specifications of the corresponding SDRAM.

5.6 DDRCTRL configuration

The DDRCTRL is a configuration of uMTCL2 with these major hardware configuration options:

– DDR mode support: DDR3/LPDDR2/LPDDR3
– Frequency ratio: 1:1
– Arbiter: include port arbiter (PA) with 2 AXI ports
– Partial write support
– CAM depth:16
– No dual HIF
– Variable priority read and variable priority write
– Dual read address queue per port

5.7 DDRCTRL registers

The DDRCTRL is configured to support DDR3/LPDDR2/LPDDR3. In the following register description section, ignore all references to other standards (mDDR, DDR2, DDR4, LPDDR4) as not being applicable.

Several register values depend on DDRCTRL configuration according to:

- Include arbiter (UMCTL2_INCL_ARB = 1)
- Partial write support (UMTCL2__PARTIAL_WR = 1)
- Variable priority read (UMCTL2_VPR_EN = 1)
- Variable priority write (UMCTL2_VPW_EN = 1)

The DDRCTRL registers are accessed only by words (32-bit).

R/W access to unmapped registers causes an APB SLVERR.

Write access to read-only registers is ignored without causing an APB SLVERR.

The Programming mode tag in the bit field describes how the field can be programmed according to:

- **Static**: the bit field is written only before DDRCTRL reset is released.
- **Dynamic**: the bit field may be written during run time.
- **Quasi-dynamic**: the bit field may be written during run time under special conditions and procedure (described in *Section 5.4.5: APB Interface (APB)*).

Quarter bus width (MSTR.data_bus_width = 10) is not supported.

For register summary see: *Table 19 on page 296*.

5.7.1 DDRCTRL master register 0 (DDRCTRL_MSTR)

Address offset: 0x000

Reset value: 0x0004 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BURST_RDWR[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLL_OFF_MODE	Res.	DATA_BUS_WIDTH[1:0]		Res.	EN_2T_TIMING_MODE	BURSTCHOP	Res.	Res.	Res.	Res.	Res.	LPDDR3	LPDDR2	Res.	DDR3
rw		rw	rw		rw	rw						rw	rw		rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **BURST_RDWR[3:0]**: SDRAM burst length used:
- 0001 - Burst length of 2 (only supported for mDDR)
- 0010 - Burst length of 4
- 0100 - Burst length of 8
- 1000 - Burst length of 16 (only supported for mDDR, LPDDR2, and LPDDR4)

All other values are reserved.

This controls the burst size used to access the SDRAM. This must match the burst length mode register setting in the SDRAM. (For BC4/8 on-the-fly mode of DDR3 and DDR4, set this field to 0x0100) The burst length of 2 is not supported with the AXI ports when MEMC_BURST_LENGTH is 8.

The burst length of 2 is only supported when the controller is operating in 1:1 frequency mode.

For DDR3, DDR4 and LPDDR3, this must be set to 0x0100 (BL8).

For LPDDR4, this must be set to 0x1000 (BL16).

Programming mode: Static

Bit 15 **DLL_OFF_MODE**: Set to 1 when the DDRCTRL and DRAM has to be put in DLL-off mode for low frequency operation.

Set to 0 to put DDRCTRL and DRAM in DLL-on mode for normal frequency operation.

If DDR4 CRC/parity retry is enabled (CRCPARCTL1.crc_parity_retry_enable = 1), dll_off_mode is not supported, and this bit must be set to '0'.

Programming mode: Quasi-dynamic Group 2

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **DATA_BUS_WIDTH[1:0]**: Selects proportion of DQ bus width that is used by the SDRAM
- 00 - Full DQ bus width to SDRAM
- 01 - Half DQ bus width to SDRAM
- 10 - Reserved
- 11 - Reserved

Programming mode: Static

Bit 11 Reserved, must be kept at reset value.

Bit 10 **EN_2T_TIMING_MODE**: If 1, then the DDRCTRL uses 2T timing. Otherwise, uses 1T timing. In 2T timing, all command signals (except chip select) are held for 2 clocks on the SDRAM bus. The chip select is asserted on the second cycle of the command

Note: 2T timing is not supported in LPDDR2/LPDDR3/LPDDR4 mode

Note: 2T timing is not supported if the configuration parameter MEMC_CMD_RTN2IDLE is set

Note: 2T timing is not supported in DDR4 geardown mode.

Note: 2T timing is not supported in Shared-AC dual channel mode and the register value is don't care.

Programming mode: Static

Bit 9 **BURSTCHOP**: When set, enable burst-chop (BC4 or 8 on-the-fly) in DDR3/DDR4. The burst-chop for Reads is exercised only in HIF configurations (UMCTL2_INCL_ARB not set) and if in full bus width mode (MSTR.data_bus_width = 00) and if MEMC_BURST_LENGTH=8 or 16. The burst-chop for writes is exercised only if partial writes enabled (UMCTL2_PARTIAL_WR=1) and if CRC is disabled (CRCPARCTL1.crc_enable = 0).

The BC4 (fixed) mode is not supported.

Programming mode: Static

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **LPDDR3**: Selects LPDDR3 SDRAM
- 1 - LPDDR3 SDRAM device in use.
- 0 - non-LPDDR3 device in use

Present only in designs configured to support LPDDR3.
Programming mode: Static

Bit 2 **LPDDR2**: Selects LPDDR2 SDRAM
- 1 - LPDDR2 SDRAM device in use.
- 0 - non-LPDDR2 device in use

Present only in designs configured to support LPDDR2.
Programming mode: Static

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DDR3**: Selects DDR3 SDRAM
- 1 - DDR3 SDRAM device in use
- 0 - non-DDR3 SDRAM device in use

Only present in designs that support DDR3.
Programming mode: Static

5.7.2 DDRCTRL operating mode status register (DDRCTRL_STAT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SELFREF_CAM_NOT_EMPTY	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_TYPE[1:0]		Res.	OPERATING_MODE[2:0]		
			r							r	r		r	r	r

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **SELFREF_CAM_NOT_EMPTY**: Self refresh with CAMs not empty. Set to 1 when Self Refresh is entered but CAMs are not drained. Cleared after exiting Self Refresh.
Programming mode: Dynamic

Bits 11:6 Reserved, must be kept at reset value.

Bits 5:4 **SELFREF_TYPE[1:0]**: Flags if Self Refresh (except LPDDR4) or SR-Powerdown (LPDDR4) is entered and if it was under Automatic Self Refresh control only or not.

- 00 - SDRAM is not in Self Refresh (except LPDDR4) or SR-Powerdown (LPDDR4). If retry is enabled by CRCPARCTRL1.crc_parity_retry_enable, this also indicates SRE command is still in parity error window or retry is in-progress.
- 11 - SDRAM is in Self Refresh (except LPDDR4) or SR-Powerdown (LPDDR4), which was caused by Automatic Self Refresh only. If retry is enabled, this guarantees SRE command is executed correctly without parity error.
- 10 - SDRAM is in Self Refresh (except LPDDR4) or SR-Powerdown (LPDDR4), which was not caused solely under Automatic Self Refresh control. It could have been caused by Hardware Low-power Interface and/or Software (PWRCTL.selfref_sw). If retry is enabled, this guarantees SRE command is executed correctly without parity error.
- 01 - SDRAM is in Self Refresh, which was caused by PHY Master Request.

Programming mode: Dynamic

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **OPERATING_MODE[2:0]**: Operating mode. This is 3-bits wide in configurations with mDDR/LPDDR2/LPDDR3/LPDDR4/DDR4 support and 2-bits in all other configurations.

Condition non-mDDR/LPDDR2/LPDDR3/LPDDR4 and non-DDR4 designs:
- 000 - Init
- 001 - Normal
- 010 - Power-down
- 011 - Self refresh

Condition mDDR/LPDDR2/LPDDR3 or DDR4 designs:
- 000 - Init
- 001 - Normal
- 010 - Power-down
- 011 - Self refresh
- 1XX - Deep power-down / Maximum Power Saving Mode

Condition LPDDR4 designs:
- 000 - Init
- 001 - Normal
- 010 - Power-down
- 011 - Self refresh / Self refresh power-down

Programming mode: Dynamic

5.7.3 DDRCTRL mode register read/write control register 0 (DDRCTRL_MRCTRL0)

Address offset: 0x010

Reset value: 0x0000 0010

Mode Register Read/Write Control Register 0.

Note: Do not enable more than one of the following fields simultaneously:

- *sw_init_int*
- *pda_en*
- *mpr_en*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR_WR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR_ADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_TYPE
rw	rw	rw	rw												rw

Bit 31 **MR_WR**: Setting this register bit to 1 triggers a mode register read or write operation. When the MR operation is complete, the DDRCTRL automatically clears this bit. The other register fields of this register must be written in a separate APB transaction, before setting this mr_wr bit. It is recommended NOT to set this signal if in Init, Deep power-down or MPSM operating modes.

Programming mode: Dynamic

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:12 **MR_ADDR[3:0]**: Address of the mode register that is to be written to.

- 0000 - MR0
- 0001 - MR1
- 0010 - MR2
- 0011 - MR3
- 0100 - MR4
- 0101 - MR5
- 0110 - MR6
- 0111 - MR7

Don't Care for LPDDR2/LPDDR3/LPDDR4 (see MRCTRL1.mr_data for mode register addressing in LPDDR2/LPDDR3/LPDDR4)

This signal is also used for writing to control words of the register chip on RDIMMs/LRDIMMs. In that case, it corresponds to the bank address bits sent to the RDIMM/LRDIMM

In case of DDR4, the bit[3:2] corresponds to the bank group bits. Therefore, the bit[3] as well as the bit[2:0] must be set to an appropriate value which is considered both the Address Mirroring of UDIMMs/RDIMMs/LRDIMMs and the Output Inversion of RDIMMs/LRDIMMs.

Programming mode: Dynamic

Bits 11:5 Reserved, must be kept at reset value.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **MR_TYPE**: Indicates whether the mode register operation is read or write. Only used for LPDDR2/LPDDR3/LPDDR4/DDR4.
- 0 - Write, MRCTRL register does not support MR read for LPDDR2/3
- 1 - Reserved

Programming mode: Dynamic

5.7.4 DDRCTRL mode register read/write control register 1 (DDRCTRL_MRCTRL1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR_DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MR_DATA[15:0]**: Mode register write data for all non-LPDDR2/non-LPDDR3/non-LPDDR4 modes.

For LPDDR2/LPDDR3/LPDDR4, MRCTRL1[15:0] are interpreted as

[15:8] MR Address

[7:0] MR data for writes, don't care for reads. This is 18-bits wide in configurations with DDR4 support and 16-bits in all other configurations.

Programming mode: Dynamic

5.7.5 DDRCTRL mode register read/write status register (DDRCTRL_MRSTAT)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_WR_BUSY
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MR_WR_BUSY**: The SoC core may initiate a MR write operation only if this signal is low. This signal goes high in the clock after the DDRCTRL accepts the MRW/MRR request. It goes low when the MRW/MRR command is issued to the SDRAM. It is recommended not to perform MRW/MRR commands when 'MRSTAT.mr_wr_busy' is high.

- 0 - Indicates that the SoC core can initiate a mode register write operation
- 1 - Indicates that mode register write operation is in progress

Programming mode: Dynamic

5.7.6 DDRCTRL temperature derate enable register (DDRCTRL_DERATEEN)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_BYTE[3:0]				Res.	DERATE_VALUE[1:0]		DERATE_ENABLE
								rw	rw	rw	rw		rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DERATE_BYTE[3:0]**: Derate byte
Present only in designs configured to support LPDDR2/LPDDR3/LPDDR4
Indicates which byte of the MRR data is used for derating. The maximum valid value depends on MEMC_DRAM_TOTAL_DATA_WIDTH.
Programming mode: Static

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **DERATE_VALUE[1:0]**: Derate value
- 0 - Derating uses +1.
- 1 - Derating uses +2.

Present only in designs configured to support LPDDR2/LPDDR3/LPDDR4
Set to 0 for all LPDDR2 speed grades as derating value of +1.875 ns is less than a core_ddrc_core_clk period.
For LPDDR3/4, if the period of core_ddrc_core_clk is less than 1.875ns, this register field should be set to 1; otherwise it should be set to 0.
Programming mode: Quasi-dynamic Group 2 and Group 4

Bit 0 **DERATE_ENABLE**: Enables derating
- 0 - Timing parameter derating is disabled
- 1 - Timing parameter derating is enabled using MR4 read value.

Present only in designs configured to support LPDDR2/LPDDR3/LPDDR4
This field must be set to '0' for non-LPDDR2/LPDDR3/LPDDR4 mode.
Programming mode: Dynamic

5.7.7 DDRCTRL temperature derate interval register (DDRCTRL_DERATEINT)

Address offset: 0x024

Reset value: 0x0080 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR4_READ_INTERVAL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR4_READ_INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MR4_READ_INTERVAL[31:0]**: Interval between two MR4 reads, used to derate the timing parameters.
Present only in designs configured to support LPDDR2/LPDDR3/LPDDR4. This register must not be set to zero.
Unit: DFI clock cycle.
Programming mode: Static

5.7.8 DDRCTRL low power control register (DDRCTRL_PWRCTL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_CAM_DRAIN_SELFREF	Res.	SELFREF_SW	Res.	EN_DFI_DRAM_CLK_DISABLE	DEEPPOWERDOWN_EN	POWERDOWN_EN	SELFREF_EN
								rw		rw		rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **DIS_CAM_DRAIN_SELFREF**: Indicates whether skipping CAM draining is allowed when entering Self-Refresh.

This register field cannot be modified while PWRCTL.selfref_sw==1.

- 0 - CAMs must be empty before entering SR
- 1 - CAMs are not emptied before entering SR (unsupported)

Note, PWRCTL.dis_cam_drain_selfref=1 is unsupported in this release. PWRCTL.dis_cam_drain_selfref=0 is required.

Programming mode: Dynamic

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SELFREF_SW**: A value of 1 to this register causes system to move to Self Refresh state immediately, as long as it is not in INIT or DPD/MPSM operating_mode. This is referred to as Software Entry/Exit to Self Refresh.

- 1 - Software Entry to Self Refresh
- 0 - Software Exit from Self Refresh

Programming mode: Dynamic

Bit 4 Reserved, must be kept at reset value.

Bit 3 **EN_DFI_DRAM_CLK_DISABLE**: Enable the assertion of dfi_dram_clk_disable whenever a clock is not required by the SDRAM.
If set to 0, dfi_dram_clk_disable is never asserted.
Assertion of dfi_dram_clk_disable is as follows:
In DDR2/DDR3, can only be asserted in Self Refresh.
In DDR4, can be asserted in following:
- in Self Refresh.
- in Maximum Power Saving Mode
In mDDR/LPDDR2/LPDDR3, can be asserted in following:
- in Self Refresh
- in Power Down
- in Deep Power Down
- during Normal operation (Clock Stop)
In LPDDR4, can be asserted in following:
- in Self Refresh Power Down
- in Power Down
- during Normal operation (Clock Stop)
Programming mode: Dynamic

Bit 2 **DEEPPOWERDOWN_EN**: When this is 1, DDRCTRL puts the SDRAM into deep power-down mode when the transaction store is empty.
This register must be reset to '0' to bring DDRCTRL out of deep power-down mode. Controller performs automatic SDRAM initialization on deep power-down exit.
Present only in designs configured to support mDDR or LPDDR2 or LPDDR3. For non-mDDR/non-LPDDR2/non-LPDDR3, this register should not be set to 1.
For performance only
Programming mode: Dynamic

Bit 1 **POWERDOWN_EN**: If true then the DDRCTRL goes into power-down after a programmable number of cycles "maximum idle clocks before power down" (PWRTMG.powerdown_to_x32).
This register bit may be re-programmed during the course of normal operation.
Programming mode: Dynamic

Bit 0 **SELFREF_EN**: If true then the DDRCTRL puts the SDRAM into Self Refresh after a programmable number of cycles "maximum idle clocks before Self Refresh (PWRTMG.selfref_to_x32)". This register bit may be re-programmed during the course of normal operation.
Programming mode: Dynamic

5.7.9 DDRCTRL low power timing register (DDRCTRL_PWRTMG)

Address offset: 0x034

Reset value: 0x0040 2010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_TO_X32[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T_DPD_X4096[7:0]								Res.	Res.	Res.	POWERDOWN_TO_X32[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SELFREF_TO_X32[7:0]**: After this many clocks of the DDRC command channel being idle the DDRCTRL automatically puts the SDRAM into Self Refresh. The DDRC command channel is considered idle when there are no HIF commands outstanding. This must be enabled in the PWRCTL.selfref_en.

Unit: Multiples of 32 DFI clocks.

For performance only.

Programming mode: Quasi-dynamic Group 4

Bits 15:8 **T_DPD_X4096[7:0]**: Minimum deep power-down time.

For mDDR, value from the JEDEC specification is 0 as mDDR exits from deep power-down mode immediately after PWRCTL.deeppowerdown_en is de-asserted.

For LPDDR2/LPDDR3, value from the JEDEC specification is 500 us.

Unit: Multiples of 4096 DFI clocks.

Present only in designs configured to support mDDR, LPDDR2 or LPDDR3.

For performance only.

Programming mode: Quasi-dynamic Group 4

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **POWERDOWN_TO_X32[4:0]**: After this many clocks of the DDRC command channel being idle the DDRCTRL automatically puts the SDRAM into power-down. The DDRC command channel is considered idle when there are no HIF commands outstanding. This must be enabled in the PWRCTL.powerdown_en.

Unit: Multiples of 32 DFI clocks

For performance only.

Programming mode: Quasi-dynamic Group 4

5.7.10 DDRCTRL hardware low power control register (DDRCTRL_HWLPCTL)

Address offset: 0x038

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HW_LP_IDLE_X32[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LP_EXIT_IDLE_EN	HW_LP_EN
														rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HW_LP_IDLE_X32[11:0]**: Hardware idle period. The cactive_ddrc output is driven low if the DDRC command channel is idle for hw_lp_idle * 32 cycles if not in INIT or DPD/MPSM operating_mode. The DDRC command channel is considered idle when there are no HIF commands outstanding. The hardware idle function is disabled when hw_lp_idle_x32=0.

Unit: Multiples of 32 DFI clocks.

For performance only.

Programming mode: Static

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **HW_LP_EXIT_IDLE_EN**: When this bit is programmed to 1 the cactive_in_ddrc pin of the DDRC can be used to exit from the automatic clock stop, automatic power down or automatic self-refresh modes. Note, it does not cause exit of Self-Refresh that is caused by Hardware Low power interface and/or software (PWRCTL.selfref_sw).

Programming mode: Static

Bit 0 **HW_LP_EN**: Enable for hardware low power interface.

Programming mode: Quasi-dynamic Group 3

5.7.11 DDRCTRL refresh control register 0 (DDRCTRL_RFSHCTL0)

Address offset: 0x050

Reset value: 0x0021 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REFRESH_MARGIN[3:0]				Res.	Res.	Res.	REFRESH_TO_X32[4]
								rw	rw	rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFRESH_TO_X32[3:0]				Res.	Res.	Res.	REFRESH_BURST[4:0]					Res.	PER_BANK_REFRESH	Res.	Res.
rw	rw	rw	rw				rw	rw	rw	rw	rw		rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **REFRESH_MARGIN[3:0]**: Threshold value in number of DFI clock cycles before the critical refresh or page timer expires. A critical refresh is to be issued before this threshold is reached. It is recommended that this not be changed from the default value, currently shown as 0x2. It must always be less than internally used t_rfc_nom/32. Note that internally used t_rfc_nom is equal to RFSHTMG.t_rfc_nom_x1_x32 * 32 if RFSHTMG.t_rfc_nom_x1_sel=0. If RFSHTMG.t_rfc_nom_x1_sel=1 (for LPDDR2/LPDDR3/LPDDR4 per-bank refresh only), internally used t_rfc_nom is equal to RFSHTMG.t_rfc_nom_x1_x32. Note that, in LPDDR2/LPDDR3/LPDDR4, internally used t_rfc_nom may be divided by four if derating is enabled (DERATEEN.derate_enable=1).

Unit: Multiples of 32 DFI clocks.

Programming mode: Dynamic - Refresh Related

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:12 **REFRESH_TO_X32[4:0]**: If the refresh timer (tRFCnom, also known as tREFI) has expired at least once, but it has not expired (RFSHCTL0.refresh_burst+1) times yet, then a speculative refresh may be performed. A speculative refresh is a refresh performed at a time when refresh would be useful, but before it is absolutely required. When the SDRAM bus is idle for a period of time determined by this RFSHCTL0.refresh_to_x32 and the refresh timer has expired at least once since the last refresh, then a speculative refresh is performed. Speculative refreshes continues successively until there are no refreshes pending or until new reads or writes are issued to the DDRCTRL.

For performance only.

Unit: Multiples of 32 DFI clocks.

Programming mode: Dynamic - Refresh Related

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:4 **REFRESH_BURST[4:0]**: The programmed value + 1 is the number of refresh timeouts that is allowed to accumulate before traffic is blocked and the refreshes are forced to execute. Closing pages to perform a refresh is a one-time penalty that must be paid for each group of refreshes. Therefore, performing refreshes in a burst reduces the per-refresh penalty of these page closings. Higher numbers for RFSHCTL.refresh_burst slightly increases utilization; lower numbers decreases the worst-case latency associated with refreshes.

- 0 - single refresh
- 1 - burst-of-2 refresh
- 7 - burst-of-8 refresh

For information on burst refresh feature refer to section 3.9 of DDR2 JEDEC specification - JESD79-2F.pdf.

For DDR2/3, the refresh is always per-rank and not per-bank. The rank refresh can be accumulated over 8*tREFI cycles using the burst refresh feature.

In DDR4 mode, according to Fine Granularity feature, 8 refreshes can be postponed in 1X mode, 16 refreshes in 2X mode and 32 refreshes in 4X mode. If using PHY-initiated updates, care must be taken in the setting of RFSHCTL0.refresh_burst, to ensure that tRFCmax is not violated due to a PHY-initiated update occurring shortly before a refresh burst was due. In this situation, the refresh burst is delayed until the PHY-initiated update is complete.

Programming mode: dynamic - refresh related

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PER_BANK_REFRESH**:

- 1 - Per bank refresh;
- 0 - All bank refresh.

Per bank refresh allows traffic to flow to other banks. Per bank refresh is not supported by all LPDDR2 devices but should be supported by all LPDDR3/LPDDR4 devices. Present only in designs configured to support LPDDR2/LPDDR3/LPDDR4

Programming mode: Static

Bits 1:0 Reserved, must be kept at reset value.

5.7.12 DDRCTRL refresh control register 3 (DDRCTRL_RFSHCTL3)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REFRESH_UPDATE_LEVEL	DIS_AUTO_REFRESH
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **REFRESH_UPDATE_LEVEL**: Toggles this signal (either from 0 to 1 or from 1 to 0) to indicate that the refresh register(s) have been updated.

refresh_update_level must not be toggled when the DDRC is in reset (core_ddrc_rstn = 0).

The refresh register(s) are automatically updated when exiting reset.

Programming mode: Dynamic

Bit 0 **DIS_AUTO_REFRESH**: When '1', disable auto-refresh generated by the DDRCTRL. When auto-refresh is disabled, the SoC core must generate refreshes using the registers DBGCMD.rankn_refresh.

When dis_auto_refresh transitions from 0 to 1, any pending refreshes are immediately scheduled by the DDRCTRL.

If DDR4 CRC/parity retry is enabled (CRCPARCTL1.crc_parity_retry_enable = 1), disable auto-refresh is not supported, and this bit must be set to '0'.

(DDR4 only) If FGR mode is enabled (RFSHCTL3.refresh_mode > 0), disable auto-refresh is not supported, and this bit must be set to '0'.

This register field is changeable on the fly.

Programming mode: Dynamic - Refresh Related

5.7.13 DDRCTRL refresh timing register (DDRCTRL_RFSHTMG)

Address offset: 0x064

Reset value: 0x0062 008C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
T_RFC_NOM_X1_SEL	Res.	Res.	Res.	T_RFC_NOM_X1_X32[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPDDR3_TREFBW_EN	Res.	Res.	Res.	Res.	Res.	T_RFC_MIN[9:0]									
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **T_RFC_NOM_X1_SEL**: Specifies whether the t_rfc_nom_x1_x32 register value is x1 or x32.
Programming mode: Dynamic - Refresh Related

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **T_RFC_NOM_X1_X32[11:0]**: tREFI: Average time interval between refreshes per rank (Specification: 7.8us for DDR2, DDR3 and DDR4. See JEDEC specification for mDDR, LPDDR2, LPDDR3 and LPDDR4).

For LPDDR2/LPDDR3/LPDDR4:

- If using all-bank refreshes (RFSHCTL0.per_bank_refresh = 0), this register should be set to tREFIab
- If using per-bank refreshes (RFSHCTL0.per_bank_refresh = 1), this register should be set to tREFIpb

When the controller is operating in 1:2 frequency ratio mode, program this to (tREFI/2), no rounding up.

In DDR4 mode, tREFI value is different depending on the refresh mode. The user should program the appropriate value from the spec based on the value programmed in the refresh mode register.

Note that if RFSHTMG.t_rfc_nom_x1_sel == 1, RFSHTMG.t_rfc_nom_x1_x32 must be greater than RFSHTMG.t_rfc_min; if RFSHTMG.t_rfc_nom_x1_sel == 0, RFSHTMG.t_rfc_nom_x1_x32 * 32 must be greater than RFSHTMG.t_rfc_min; RFSHTMG.t_rfc_nom_x1_x32 must be greater than 0x1.

- Non-DDR4 or DDR4 Fixed 1x mode: RFSHTMG.t_rfc_nom_x1_x32 must be less than or equal to 0xFFE.
- DDR4 Fixed 2x mode: RFSHTMG.t_rfc_nom_x1_x32 must be less than or equal to 0x7FF.
- DDR4 Fixed 4x mode: RFSHTMG.t_rfc_nom_x1_x32 must be less than or equal to 0x3FF.

Unit: Clocks or multiples of 32 clocks, depending on RFSHTMG.t_rfc_nom_x1_sel.

Programming mode: Dynamic - Refresh Related

Bit 15 **LPDDR3_TREFBW_EN**: Used only when LPDDR3 memory type is connected. Should only be changed when DDRCTRL is in reset. Specifies whether to use the tREFBW parameter (required by some LPDDR3 devices which comply with earlier versions of the LPDDR3 JEDEC specification) or not:

- 0 - tREFBW parameter not used
- 1 - tREFBW parameter used

Programming mode: Static

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:0 **T_RFC_MIN[9:0]**: tRFC (min): Minimum time from refresh to refresh or activate.

When the controller is operating in 1:1 mode, t_rfc_min should be set to RoundUp(tRFCmin/tCK).

When the controller is operating in 1:2 mode, t_rfc_min should be set to RoundUp(RoundUp(tRFCmin/tCK)/2).

In LPDDR2/LPDDR3/LPDDR4 mode:

- If using all-bank refreshes, the tRFCmin value in the above equations is equal to tRFCab
- If using per-bank refreshes, the tRFCmin value in the above equations is equal to tRFCpb

In DDR4 mode, the tRFCmin value in the above equations is different depending on the refresh mode (fixed 1X,2X,4X) and the device density. The user should program the appropriate value from the spec based on the 'refresh_mode' and the device density that is used.

Unit: Clocks.

Programming mode: Dynamic - Refresh Related

5.7.14 DDRCTRL CRC parity control register 0 (DDRCTRL_CRCPARCTL0)

Address offset: 0x0C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ALERT_ERR_CNT_CLR	DFI_ALERT_ERR_INT_CLR	DFI_ALERT_ERR_INT_EN
													rc_w1	rc_w1	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DFI_ALERT_ERR_CNT_CLR**: DFI alert error count clear. Clear bit for DFI alert error counter. Asserting this bit, clears the DFI alert error counter, CRCPARSTAT.dfi_alert_err_cnt. When the clear operation is complete, the DDRCTRL automatically clears this bit.

Programming mode: Dynamic

Bit 1 **DFI_ALERT_ERR_INT_CLR**: Interrupt clear bit for DFI alert error. If this bit is set, the alert error interrupt on CRCPARSTAT.dfi_alert_err_int is cleared. When the clear operation is complete, the DDRCTRL automatically clears this bit.
Programming mode: Dynamic

Bit 0 **DFI_ALERT_ERR_INT_EN**: Interrupt enable bit for DFI alert error. If this bit is set, any parity/CRC error detected on the dfi_alert_n input results in an interrupt being set on CRCPARSTAT.dfi_alert_err_int.
Programming mode: Dynamic

5.7.15 DDRCTRL CRC parity status register (DDRCTRL_CRCPARSTAT)

Address offset: 0x0CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ALERT_ERR_INT
															r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DFI_ALERT_ERR_CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DFI_ALERT_ERR_INT**: DFI alert error interrupt.
If a parity/CRC error is detected on dfi_alert_n, and the interrupt is enabled by CRCPARCTL0.dfi_alert_err_int_en, this interrupt bit is set. It remains set until cleared by CRCPARCTL0.dfi_alert_err_int_clr
Programming mode: Static

Bits 15:0 **DFI_ALERT_ERR_CNT[15:0]**: DFI alert error count.
If a parity/CRC error is detected on dfi_alert_n, this counter be incremented. This is independent of the setting of CRCPARCTL0.dfi_alert_err_int_en. It saturates at 0xFFFF, and can be cleared by asserting CRCPARCTL0.dfi_alert_err_cnt_clr.
Programming mode: Static

5.7.16 DDRCTRL SDRAM initialization register 0 (DDRCTRL_INIT0)

Address offset: 0x0D0

Reset value: 0x0002 004E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SKIP_DRAM_INIT[1:0]		Res.	Res.	Res.	Res.	POST_CKE_X1024[9:0]									
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	PRE_CKE_X1024[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **SKIP_DRAM_INIT[1:0]**: If lower bit is enabled the SDRAM initialization routine is skipped. The upper bit decides what state the controller starts up in when reset is removed

- 00 - SDRAM Initialization routine is run after power-up
- 01 - SDRAM Initialization routine is skipped after power-up. Controller starts up in Normal Mode
- 11 - SDRAM Initialization routine is skipped after power-up. Controller starts up in Self-refresh Mode
- 10 - Reserved.

Programming mode: Quasi-dynamic Group 2

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:16 **POST_CKE_X1024[9:0]**: Cycles to wait after driving CKE high to start the SDRAM initialization sequence.

Unit: 1024 DFI clock cycles.

DDR2 typically requires a 400 ns delay, requiring this value to be programmed to 2 at all clock speeds.

LPDDR2/LPDDR3 typically requires this to be programmed for a delay of 200 us.

LPDDR4 typically requires this to be programmed for a delay of 2 us.

When the controller is operating in 1:2 frequency ratio mode, program this to JEDEC spec value divided by 2, and round it up to the next integer value.

Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **PRE_CKE_X1024[11:0]**: Cycles to wait after reset before driving CKE high to start the SDRAM initialization sequence.

Unit: 1024 DFI clock cycles.

DDR2 specifications typically require this to be programmed for a delay of >= 200 us.

LPDDR2/LPDDR3: tINIT1 of 100 ns (min)

LPDDR4: tINIT3 of 2 ms (min)

When the controller is operating in 1:2 frequency ratio mode, program this to JEDEC spec value divided by 2, and round it up to the next integer value.

For DDR3/DDR4 RDIMMs, this should include the time needed to satisfy tSTAB

Programming mode: Static

5.7.17 DDRCTRL SDRAM initialization register 1 (DDRCTRL_INIT1)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAM_RSTN_X1024[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE_OCD_X32[3:0]			
												rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **DRAM_RSTN_X1024[8:0]**: Number of cycles to assert SDRAM reset signal during init sequence.

This is only present for designs supporting DDR3, DDR4 or LPDDR4 devices. For use with a DDR PHY, this should be set to a minimum of 1.

When the controller is operating in 1:2 frequency ratio mode, program this to JEDEC spec value divided by 2, and round it up to the next integer value.

Unit: 1024 DFI clock cycles.

Programming mode: Static

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **PRE_OCD_X32[3:0]**: Wait period before driving the OCD complete command to SDRAM.

Unit: Counts of a global timer that pulses every 32 DFI clock cycles.

There is no known specific requirement for this; it may be set to zero.

Programming mode: Static

5.7.18 DDRCTRL SDRAM initialization register 2 (DDRCTRL_INIT2)

Address offset: 0x0D8

Reset value: 0x0000 0D05

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IDLE_AFTER_RESET_X32[7:0]								Res.	Res.	Res.	Res.	MIN_STABLE_CLOCK_X1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **IDLE_AFTER_RESET_X32[7:0]**: Idle time after the reset command, tINIT4. Present only in designs configured to support LPDDR2.
When the controller is operating in 1:2 frequency ratio mode, program this to JEDEC spec value divided by 2, and round it up to the next integer value.
Unit: 32 DFI clock cycles.
Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **MIN_STABLE_CLOCK_X1[3:0]**: Time to wait after the first CKE high, tINIT2. Present only in designs configured to support LPDDR2/LPDDR3.
LPDDR2/LPDDR3 typically requires 5 x tCK delay.
When the controller is operating in 1:2 frequency ratio mode, program this to JEDEC spec value divided by 2, and round it up to the next integer value.
Unit: DFI clock cycles.
Programming mode: Static

5.7.19 DDRCTRL SDRAM initialization register 3 (DDRCTRL_INIT3)

Address offset: 0x0DC

Reset value: 0x0000 0510

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **MR[15:0]**: DDR2: Value to write to MR register. Bit 8 is for DLL and the setting here is ignored. The DDRCTRL sets this bit appropriately.
DDR3/DDR4: Value loaded into MR0 register.
mDDR: Value to write to MR register.
LPDDR2/LPDDR3/LPDDR4 - Value to write to MR1 register
Programming mode: Quasi-dynamic Group 1 and Group 4

Bits 15:0 **EMR[15:0]**: DDR2: Value to write to EMR register. Bits 9:7 are for OCD and the setting in this register is ignored. The DDRCTRL sets those bits appropriately.
DDR3/DDR4: Value to write to MR1 register Set bit 7 to 0. If PHY-evaluation mode training is enabled, this bit is set appropriately by the DDRCTRL during write leveling.
mDDR: Value to write to EMR register.
LPDDR2/LPDDR3/LPDDR4 - Value to write to MR2 register
Programming mode: Quasi-dynamic Group 4

5.7.20 DDRCTRL SDRAM initialization register 4 (DDRCTRL_INIT4)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EMR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **EMR2[15:0]**: DDR2: Value to write to EMR2 register.
DDR3/DDR4: Value to write to MR2 register
LPDDR2/LPDDR3/LPDDR4: Value to write to MR3 register
mDDR: Unused
Programming mode: Quasi-dynamic Group 4

Bits 15:0 **EMR3[15:0]**: DDR2: Value to write to EMR3 register.
DDR3/DDR4: Value to write to MR3 register
mDDR/LPDDR2/LPDDR3: Unused
LPDDR4: Value to write to MR13 register
Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.21 DDRCTRL SDRAM initialization register 5 (DDRCTRL_INIT5)

Address offset: 0x0E4

Reset value: 0x0010 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ZQINIT_X32[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	MAX_AUTO_INIT_X1024[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DEV_ZQINIT_X32[7:0]**: ZQ initial calibration, tZQINIT. Present only in designs configured to support DDR3 or DDR4 or LPDDR2/LPDDR3.
DDR3 typically requires 512 SDRAM clock cycles.
DDR4 requires 1024 SDRAM clock cycles.
LPDDR2/LPDDR3 requires 1 us.
When the controller is operating in 1:2 frequency ratio mode, program this to JEDEC spec value divided by 2, and round it up to the next integer value.
Unit: 32 DFI clock cycles.
Programming mode: Static

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **MAX_AUTO_INIT_X1024[9:0]**: Maximum duration of the auto initialization, tINIT5. Present only in designs configured to support LPDDR2/LPDDR3.
LPDDR2/LPDDR3 typically requires 10 us.
Unit: 1024 DFI clock cycles.
Programming mode: Static

5.7.22 DDRCTRL DIMM control register (DDRCTRL_DIMMCTL)

Address offset: 0x0F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIMM_ADDR_MIRR_EN	DIMM_STAGGER_CS_EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DIMM_ADDR_MIRR_EN**: Address Mirroring Enable (for multi-rank UDIMM implementations and multi-rank DDR4 RDIMM/LRDIMM implementations).
Some UDIMMs and DDR4 RDIMMs/LRDIMMs implement address mirroring for odd ranks, which means that the following address, bank address and bank group bits are swapped: (A3, A4), (A5, A6), (A7, A8), (BA0, BA1) and also (A11, A13), (BG0, BG1) for the DDR4. Setting this bit ensures that, for mode register accesses during the automatic initialization routine, these bits are swapped within the DDRCTRL to compensate for this UDIMM/RDIMM/LRDIMM swapping. In addition to the automatic initialization routine, in case of DDR4 UDIMM/RDIMM/LRDIMM, they are swapped during the automatic MRS access to enable/disable of a particular DDR4 feature.
Note: This has no effect on the address of any other memory accesses, or of software-driven mode register accesses.
This is not supported for mDDR, LPDDR2, LPDDR3 or LPDDR4 SDRAMs.
Note: In case of x16 DDR4 DIMMs, BG1 output of MRS for the odd ranks is same as BG0 because BG1 is invalid, hence dimm_dis_bg_mirroring register must be set to 1.
- 1 - For odd ranks, implement address mirroring for MRS commands to during initialization and for any automatic DDR4 MRS commands (to be used if UDIMM/RDIMM/LRDIMM implements address mirroring)
- 0 - Do not implement address mirroring

Programming mode: Static

Bit 0 **DIMM_STAGGER_CS_EN**: Staggering enable for multi-rank accesses (for multi-rank UDIMM, RDIMM and LRDIMM implementations only). This is not supported for mDDR, LPDDR2, LPDDR3 or LPDDR4 SDRAMs.

Note: Even if this bit is set it does not take care of software driven MR commands (via MRCTRL0/MRCTRL1), where software is responsible to send them to separate ranks as appropriate.

- 1 - (DDR4) Send MRS commands to each ranks separately
- 1 - (non-DDR4) Send all commands to even and odd ranks separately
- 0 - Do not stagger accesses

Programming mode: Static.

5.7.23 DDRCTRL SDRAM timing register 0 (DDRCTRL_DRAMTMG0)

Address offset: 0x100

Reset value: 0x0F10 1B0F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WR2PRE[6:0]							Res.	Res.	T_FAW[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	T_RAS_MAX[6:0]							Res.	Res.	T_RAS_MIN[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **WR2PRE[6:0]**:

Minimum time between write and precharge to same bank.

Unit: Clocks

Specifications: WL + BL/2 + tWR = approximately 8 cycles + 15 ns = 14 clocks @400MHz and less for lower frequencies

where:

- WL = write latency
- BL = burst length. This must match the value programmed in the BL bit of the mode register to the SDRAM. BST (burst terminate) is not supported at present.
- tWR = Write recovery time. This comes directly from the SDRAM specification.

Add one extra cycle for LPDDR2/LPDDR3/LPDDR4 for this parameter.

When the controller is operating in 1:2 frequency ratio mode, 1T mode, divide the above value by 2. No rounding up.

When the controller is operating in 1:2 frequency ratio mode, 2T mode or LPDDR4 mode, divide the above value by 2 and round it up to the next integer value.

Note that, depending on the PHY, if using LRDIMM, it may be necessary to adjust the value of this parameter to compensate for the extra cycle of latency through the LRDIMM.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **T_FAW[5:0]**: tFAW Valid only when 8 or more banks(or banks x bank groups) are present.
In 8-bank design, at most 4 banks must be activated in a rolling window of tFAW cycles.
When the controller is operating in 1:2 frequency ratio mode, program this to (tFAW/2) and round up to next integer value.
In a 4-bank design, set this register to 0x1 independent of the 1:1/1:2 frequency mode.
Unit: Clocks
Programming mode: Quasi-dynamic Group 2 and Group 4.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **T_RAS_MAX[6:0]**: tRAS(max): Maximum time between activate and precharge to same bank. This is the maximum time that a page can be kept open
Minimum value of this register is 1. Zero is invalid.
When the controller is operating in 1:2 frequency ratio mode, program this to (tRAS(max)-1)/2. No rounding up.
Unit: Multiples of 1024 clocks.
Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **T_RAS_MIN[5:0]**: tRAS(min): Minimum time between activate and precharge to the same bank.
When the controller is operating in 1:2 frequency mode, 1T mode, program this to tRAS(min)/2. No rounding up.
When the controller is operating in 1:2 frequency ratio mode, 2T mode or LPDDR4 mode, program this to (tRAS(min)/2) and round it up to the next integer value.
Unit: Clocks.
Programming mode: Quasi-dynamic Group 2 and Group 4.

5.7.24 DDRCTRL SDRAM timing register 1 (DDRCTRL_DRAMTMG1)

Address offset: 0x104

Reset value: 0x0008 0414

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XP[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RD2PRE[5:0]						Res.	T_RC[6:0]						
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **T_XP[4:0]**: tXP: Minimum time after power-down exit to any operation. For DDR3, this should be programmed to tXPDLL if slow powerdown exit is selected in MR0[12].
If C/A parity for DDR4 is used, set to (tXP+PL) instead.
If LPDDR4 is selected and its spec has tCKELPD parameter, set to the larger of tXP and tCKELPD instead.
When the controller is operating in 1:2 frequency ratio mode, program this to (tXP/2) and round it up to the next integer value.
Units: Clocks.
Programming mode: Quasi-dynamic Group 2 and Group 4.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **RD2PRE[5:0]**: tRTP: Minimum time from read to precharge of same bank.
- DDR2: tAL + BL/2 + max(tRTP, 2) - 2
- DDR3: tAL + max (tRTP, 4)
- DDR4: Max of following two equations:
  tAL + max (tRTP, 4) or,
  RL + BL/2 - tRP (*).
- mDDR: BL/2
- LPDDR2: Depends on if it's LPDDR2-S2 or LPDDR2-S4:
  LPDDR2-S2: BL/2 + tRTP - 1.
  LPDDR2-S4: BL/2 + max(tRTP,2) - 2.
- LPDDR3: BL/2 + max(tRTP,4) - 4
- LPDDR4: BL/2 + max(tRTP,8) - 8

(*) When both DDR4 SDRAM and ST-MRAM are used simultaneously, use SDRAM's tRP value for calculation.

When the controller is operating in 1:2 mode, 1T mode, divide the above value by 2. No rounding up.

When the controller is operating in 1:2 mode, 2T mode or LPDDR4 mode, divide the above value by 2 and round it up to the next integer value.

Unit: Clocks.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **T_RC[6:0]**: tRC: Minimum time between activates to same bank.

When the controller is operating in 1:2 frequency ratio mode, program this to (tRC/2) and round up to next integer value.

Unit: Clocks.

Programming mode: Quasi-dynamic Group 2 and Group 4.

5.7.25 DDRCTRL SDRAM timing register 2 (DDRCTRL_DRAMTMG2)

Address offset: 0x108

Reset value: 0x0305 060D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	WRITE_LATENCY[5:0]						Res.	Res.	READ_LATENCY[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	RD2WR[5:0]						Res.	Res.	WR2RD[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **WRITE_LATENCY[5:0]**: Set to WL

The time from write command to write data on SDRAM interface. This must be set to WL.

For mDDR, it should normally be set to 1.

Note that, depending on the PHY, if using RDIMM/LRDIMM, it may be necessary to adjust the value of WL to compensate for the extra cycle of latency through the RDIMM/LRDIMM.

When the controller is operating in 1:2 frequency ratio mode, divide the value calculated using the above equation by 2, and round it up to next integer.

This register field is not required for DDR2 and DDR3 (except if MEMC_TRAINING is set), as the DFI read and write latencies defined in DFITMG0 and DFITMG1 are sufficient for those protocols.

Unit: clocks.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **READ_LATENCY[5:0]**: Set to RL

The time from read command to read data on SDRAM interface. This must be set to RL.

Note that, depending on the PHY, if using RDIMM/LRDIMM, it may be necessary to adjust the value of RL to compensate for the extra cycle of latency through the RDIMM/LRDIMM.

When the controller is operating in 1:2 frequency ratio mode, divide the value calculated using the above equation by 2, and round it up to next integer.

This register field is not required for DDR2 and DDR3 (except if MEMC_TRAINING is set), as the DFI read and write latencies defined in DFITMG0 and DFITMG1 are sufficient for those protocols.

Unit: clocks.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **RD2WR[5:0]**:

DDR2/3/mDDR: RL + BL/2 + 2 - WL

DDR4: RL + BL/2 + 1 + WR_PREAMBLE - WL

LPDDR2/LPDDR3: RL + BL/2 + RU(tDQSCKmax/tCK) + 1 - WL

LPDDR4(DQ ODT is Disabled): RL + BL/2 + RU(tDQSCKmax/tCK) + WR_PREAMBLE + RD_POSTAMBLE - WL

LPDDR4(DQ ODT is Enabled) : RL + BL/2 + RU(tDQSCKmax/tCK) + RD_POSTAMBLE - ODTLon - RU(tODTon(min)/tCK)

Minimum time from read command to write command. Include time for bus turnaround and all per-bank, per-rank, and global constraints. Please see the relevant PHY databook for details of what should be included here.

Unit: Clocks.

Where:

- WL = write latency
- BL = burst length. This must match the value programmed in the BL bit of the mode register to the SDRAM
- RL = read latency = CAS latency
- WR_PREAMBLE = write preamble. This is unique to DDR4 and LPDDR4.
- RD_POSTAMBLE = read postamble. This is unique to LPDDR4.

For LPDDR2/LPDDR3/LPDDR4, if derating is enabled (DERATEEN.derate_enable=1), derated tDQSCKmax should be used.

When the controller is operating in 1:2 frequency ratio mode, divide the value calculated using the above equation by 2, and round it up to next integer.

Note that, depending on the PHY, if using LRDIMM, it may be necessary to adjust the value of this parameter to compensate for the extra cycle of latency through the LRDIMM.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **WR2RD[5:0]**:

DDR4: CWL + PL + BL/2 + tWTR_L

LPDDR2/3/4: WL + BL/2 + tWTR + 1

Others: CWL + BL/2 + tWTR

In DDR4, minimum time from write command to read command for same bank group. In others, minimum time from write command to read command. Includes time for bus turnaround, recovery times, and all per-bank, per-rank, and global constraints.

Unit: Clocks.

Where:

- CWL = CAS write latency
- WL = Write latency
- PL = Parity latency
- BL = burst length. This must match the value programmed in the BL bit of the mode register to the SDRAM
- tWTR_L = internal write to read command delay for same bank group. This comes directly from the SDRAM specification.
- tWTR = internal write to read command delay. This comes directly from the SDRAM specification.

Add one extra cycle for LPDDR2/LPDDR3/LPDDR4 operation.

When the controller is operating in 1:2 mode, divide the value calculated using the above equation by 2, and round it up to next integer.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4

5.7.26 DDRCTRL SDRAM timing register 3 (DDRCTRL_DRAMTMG3)

Address offset: 0x10C

Reset value: 0x0050 400C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	T_MRW[9:0]										Res.	Res.	T_MRD[5:4]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
T_MRD[3:0]				Res.	Res.	T_MOD[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:20 **T_MRW[9:0]**: Time to wait after a mode register write or read (MRW or MRR).
Present only in designs configured to support LPDDR2, LPDDR3 or LPDDR4.
LPDDR2 typically requires value of 5.
LPDDR3 typically requires value of 10.
LPDDR4: Set this to the larger of tMRW and tMRWCKEL.
For LPDDR2, this register is used for the time from a MRW/MRR to all other commands.
When the controller is operating in 1:2 frequency ratio mode, program this to the above values divided by 2 and round it up to the next integer value.
For LDPDR3, this register is used for the time from a MRW/MRR to a MRW/MRR.
Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:12 **T_MRD[5:0]**: tMRD: Cycles to wait after a mode register write or read. Depending on the connected SDRAM, tMRD represents:
DDR2/mDDR: Time from MRS to any command
DDR3/4: Time from MRS to MRS command
LPDDR2: not used
LPDDR3/4: Time from MRS to non-MRS command.
When the controller is operating in 1:2 frequency ratio mode, program this to (tMRD/2) and round it up to the next integer value.
If C/A parity for DDR4 is used, set to tMRD_PAR(tMOD+PL) instead.
Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **T_MOD[9:0]**: tMOD: Parameter used only in DDR3 and DDR4. Cycles between load mode command and following non-load mode command.

If C/A parity for DDR4 is used, set to tMOD_PAR(tMOD+PL) instead.

If MPR writes for DDR4 are used, set to tMOD + AL (or tMPD_PAR + AL if C/A parity is also used).

Set to tMOD if controller is operating in 1:1 frequency ratio mode, or tMOD/2 (rounded up to next integer) if controller is operating in 1:2 frequency ratio mode. Note that if using RDIMM/LRDIMM, depending on the PHY, it may be necessary to adjust the value of this parameter to compensate for the extra cycle of latency applied to mode register writes by the RDIMM/LRDIMM chip.

Also note that if using LRDIMM, the minimum value of this register is tMRD_L2 if controller is operating in 1:1 frequency ratio mode, or tMRD_L2/2 (rounded up to next integer) if controller is operating in 1:2 frequency ratio mode.

Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.27 DDRCTRL SDRAM timing register 4 (DDRCTRL_DRAMTMG4)

Address offset: 0x110

Reset value: 0x0504 0405

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	T_RCD[4:0]					Res.	Res.	Res.	Res.	T_CCD[3:0]			
			rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	T_RRD[3:0]				Res.	Res.	Res.	T_RP[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **T_RCD[4:0]**: tRCD - tAL: Minimum time from activate to read or write command to same bank.

When the controller is operating in 1:2 frequency ratio mode, program this to ((tRCD - tAL)/2) and round it up to the next integer value.

Minimum value allowed for this register is 1, which implies minimum (tRCD - tAL) value to be 2 when the controller is operating in 1:2 frequency ratio mode.

Unit: Clocks.

Programming mode: Quasi-dynamic Group 1 and Group 2 and Group 4

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **T_CCD[3:0]**: DDR4: tCCD_L: This is the minimum time between two reads or two writes for same bank group.

Others: tCCD: This is the minimum time between two reads or two writes.

When the controller is operating in 1:2 frequency ratio mode, program this to (tCCD_L/2 or tCCD/2) and round it up to the next integer value.

Unit: clocks.

Programming mode: Quasi-dynamic Group 2 and Group 4.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **T_RRD[3:0]**: DDR4: tRRD_L: Minimum time between activates from bank "a" to bank "b" for same bank group.

Others: tRRD: Minimum time between activates from bank "a" to bank "b"

When the controller is operating in 1:2 frequency ratio mode, program this to (tRRD_L/2 or tRRD/2) and round it up to the next integer value.

Unit: Clocks.

Programming mode: Quasi-dynamic Group 2 and Group 4.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **T_RP[4:0]**: tRP: Minimum time from precharge to activate of same bank.

When the controller is operating in 1:1 frequency ratio mode, t_rp should be set to RoundUp(tRP/tCK).

When the controller is operating in 1:2 frequency ratio mode, t_rp should be set to RoundDown(RoundUp(tRP/tCK)/2) + 1.

When the controller is operating in 1:2 frequency ratio mode in LPDDR4, t_rp should be set to RoundUp(RoundUp(tRP/tCK)/2).

Unit: Clocks.

Programming mode: Quasi-dynamic Group 2 and Group 4.

5.7.28 DDRCTRL SDRAM timing register 5 (DDRCTRL_DRAMTMG5)

Address offset: 0x114

Reset value: 0x0505 0403

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	T_CKSRX[3:0]				Res.	Res.	Res.	Res.	T_CKSRE[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	T_CKESR[5:0]						Res.	Res.	Res.	T_CKE[4:0]				
		rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **T_CKSRX[3:0]**: This is the time before Self Refresh Exit that CK is maintained as a valid clock before issuing SRX. Specifies the clock stable time before SRX.

Recommended settings:

- mDDR: 1
- LPDDR2: 2
- LPDDR3: 2
- LPDDR4: tCKCKEH
- DDR2: 1
- DDR3: tCKSRX
- DDR4: tCKSRX

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **T_CKSRE[3:0]**: This is the time after Self Refresh Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after SRE.

Recommended settings:

- mDDR: 0
- LPDDR2: 2
- LPDDR3: 2
- LPDDR4: tCKELCK
- DDR2: 1
- DDR3: max (10 ns, 5 tCK)
- DDR4: max (10 ns, 5 tCK) (+ PL(parity latency)(*))

(*)Only if CRCPARCTL1.caparity_disable_before_sr=0, this register should be increased by PL.

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **T_CKESR[5:0]**: Minimum CKE low width for Self refresh or Self refresh power down entry to exit timing in memory clock cycles.

Recommended settings:

- mDDR: tRFC
- LPDDR2: tCKESR
- LPDDR3: tCKESR
- LPDDR4: max(tCKE, tSR)
- DDR2: tCKE
- DDR3: tCKE + 1
- DDR4: tCKE + 1 (+ PL(parity latency)(*))

(*)Only if CRCPARCTL1.caparity_disable_before_sr=0, this register should be increased by PL.

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **T_CKE[4:0]**: Minimum number of cycles of CKE HIGH/LOW during power-down and self refresh.

- LPDDR2/LPDDR3 mode: Set this to the larger of tCKE or tCKESR
- LPDDR4 mode: Set this to the larger of tCKE or tSR.
- Non-LPDDR2/non-LPDDR3/non-LPDDR4 designs: Set this to tCKE value.

When the controller is operating in 1:2 frequency ratio mode, program this to (value described above)/2 and round it up to the next integer value.

Unit: Clocks.

Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.29 DDRCTRL SDRAM timing register 6 (DDRCTRL_DRAMTMG6)

Address offset: 0x118

Reset value: 0x0202 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	T_CKDPDE[3:0]				Res.	Res.	Res.	Res.	T_CKDPDX[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CKCSX[3:0]			
												rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **T_CKDPDE[3:0]**: This is the time after Deep Power Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after DPDE.

Recommended settings:

- mDDR: 0
- LPDDR2: 2
- LPDDR3: 2

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

This is only present for designs supporting mDDR or LPDDR2/LPDDR3 devices.

Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **T_CKDPDX[3:0]**: This is the time before Deep Power Down Exit that CK is maintained as a valid clock before issuing DPDX. Specifies the clock stable time before DPDX.

Recommended settings:

- mDDR: 1
- LPDDR2: 2
- LPDDR3: 2

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

This is only present for designs supporting mDDR or LPDDR2 devices.

Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **T_CKCSX[3:0]**: This is the time before Clock Stop Exit that CK is maintained as a valid clock before issuing Clock Stop Exit. Specifies the clock stable time before next command after Clock Stop Exit.

Recommended settings:

- mDDR: 1
- LPDDR2: tXP + 2
- LPDDR3: tXP + 2
- LPDDR4: tXP + 2

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

This is only present for designs supporting mDDR or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.30 DDRCTRL SDRAM timing register 7 (DDRCTRL_DRAMTMG7)

Address offset: 0x11C

Reset value: 0x0000 0202

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	T_CKPDE[3:0]				Res.	Res.	Res.	Res.	T_CKPDX[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **T_CKPDE[3:0]**: This is the time after Power Down Entry that CK is maintained as a valid clock. Specifies the clock disable delay after PDE.

Recommended settings:

- mDDR: 0
- LPDDR2: 2
- LPDDR3: 2
- LPDDR4: tCKELCK

When using DDR2/3/4 SDRAM, this register should be set to the same value as DRAMTMG5.t_cksre.

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

This is only present for designs supporting mDDR or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Quasi-dynamic Group 2 and Group 4

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **T_CKPDX[3:0]**: This is the time before Power Down Exit that CK is maintained as a valid clock before issuing PDX. Specifies the clock stable time before PDX.

Recommended settings:

- mDDR: 0
- LPDDR2: 2
- LPDDR3: 2
- LPDDR4: 2

When using DDR2/3/4 SDRAM, this register should be set to the same value as DRAMTMG5.t_cksrx.

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

This is only present for designs supporting mDDR or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.31 DDRCTRL SDRAM timing register 8 (DDRCTRL_DRAMTMG8)

Address offset: 0x120

Reset value: 0x0000 4405

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	T_XS_DLL_X32[6:0]							Res.	T_XS_X32[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **T_XS_DLL_X32[6:0]**: tXSDLL: Exit Self Refresh to the commands requiring a locked DLL.
When the controller is operating in 1:2 frequency ratio mode, program this to the above value divided by 2 and round up to next integer value.
Unit: Multiples of 32 clocks.
Note: Used only for DDR2, DDR3 and DDR4 SDRAMs.
Programming mode: Quasi-dynamic Group 2 and Group 4

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **T_XS_X32[6:0]**: tXS: Exit Self Refresh to commands not requiring a locked DLL.
When the controller is operating in 1:2 frequency ratio mode, program this to the above value divided by 2 and round up to next integer value.
Unit: Multiples of 32 clocks.
Note: Used only for DDR2, DDR3 and DDR4 SDRAMs.
Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.32 DDRCTRL SDRAM timing register 14 (DDRCTRL_DRAMTMG14)

Address offset: 0x138

Reset value: 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	T_XSR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **T_XSR[11:0]**: tXSR: Exit Self Refresh to any command.
When the controller is operating in 1:2 frequency ratio mode, program this to the above value divided by 2 and round up to next integer value.
Note: Used only for mDDR/LPDDR2/LPDDR3/LPDDR4 mode.
Programming mode: Quasi-dynamic Group 2 and Group 4

5.7.33 DDRCTRL SDRAM timing register 15 (DDRCTRL_DRAMTMG15)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_DFI_LP_T_STAB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_STAB_X32[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EN_DFI_LP_T_STAB**:

- 1 - Enable using tSTAB when exiting DFI LP. Needs to be set when the PHY is stopping the clock during DFI LP to save maximum power.
- 0 - Disable using tSTAB when exiting DFI LP

Programming mode: Quasi-dynamic Group 2 and Group 4.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **T_STAB_X32[7:0]**: tSTAB: Stabilization time.

It is required in the following two cases for DDR3/DDR4 RDIMM:

- When exiting power saving mode, if the clock was stopped, after re-enabling it the clock must be stable for a time specified by tSTAB
- In the case of input clock frequency change (DDR4)
- After issuing control words that refers to clock timing (Specification: 6us for DDR3, 5us for DDR4).

When the controller is operating in 1:2 frequency ratio mode, program this to recommended value divided by two and round it up to next integer.

Unit: Multiples of 32 clock cycles.

Programming mode: Quasi-dynamic Group 2 and Group 4.

5.7.34 DDRCTRL ZQ control register 0 (DDRCTRL_ZQCTL0)

Address offset: 0x180

Reset value: 0x0200 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS_AUTO_ZQ	DIS_SRX_ZQCL	ZQ_RESISTOR_SHARED	Res.	Res.	T_ZQ_LONG_NOP[10:0]										
rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	T_ZQ_SHORT_NOP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DIS_AUTO_ZQ**:

- 1 - Disable DDRCTRL generation of ZQCS/MPC(ZQ calibration) command. Register DBGCMD.zq_calib_short can be used instead to issue ZQ calibration request from APB module.
- 0 - Internally generate ZQCS/MPC(ZQ calibration) commands based on ZQCTL1.t_zq_short_interval_x1024.

This is only present for designs supporting DDR3/DDR4 or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Dynamic

Bit 30 **DIS_SRX_ZQCL**:

- 1 - Disable issuing of ZQCL/MPC(ZQ calibration) command at Self-Refresh/SR-Powerdown exit. Only applicable when run in DDR3 or DDR4 or LPDDR2 or LPDDR3 or LPDDR4 mode.
- 0 - Enable issuing of ZQCL/MPC(ZQ calibration) command at Self-Refresh/SR-Powerdown exit. Only applicable when run in DDR3 or DDR4 or LPDDR2 or LPDDR3 or LPDDR4 mode.

This is only present for designs supporting DDR3/DDR4 or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Quasi-dynamic Group 2 and Group 4.

Bit 29 **ZQ_RESISTOR_SHARED**:

- 1 - Denotes that ZQ resistor is shared between ranks. Means ZQinit/ZQCL/ZQCS/MPC(ZQ calibration) commands are sent to one rank at a time with tZQinit/tZQCL/tZQCS/tZQCAL/tZQLAT timing met between commands so that commands to different ranks do not overlap.
- 0 - ZQ resistor is not shared.

This is only present for designs supporting DDR3/DDR4 or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Static.

Bits 28:27 Reserved, must be kept at reset value.

Bits 26:16 **T_ZQ_LONG_NOP[10:0]**: tZQoper for DDR3/DDR4, tZQCL for LPDDR2/LPDDR3, tZQCAL for LPDDR4: Number of DFI clock cycles of NOP required after a ZQCL (ZQ calibration long)/MPC(ZQ Start) command is issued to SDRAM.

When the controller is operating in 1:2 frequency ratio mode:

DDR3/DDR4: program this to tZQoper/2 and round it up to the next integer value.

LPDDR2/LPDDR3: program this to tZQCL/2 and round it up to the next integer value.

LPDDR4: program this to tZQCAL/2 and round it up to the next integer value.

This is only present for designs supporting DDR3/DDR4 or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Static.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **T_ZQ_SHORT_NOP[9:0]**: tZQCS for DDR3/DD4/LPDDR2/LPDDR3, tZQLAT for LPDDR4: Number of DFI clock cycles of NOP required after a ZQCS (ZQ calibration short)/MPC(ZQ Latch) command is issued to SDRAM.

When the controller is operating in 1:2 frequency ratio mode, program this to tZQCS/2 and round it up to the next integer value.

This is only present for designs supporting DDR3/DDR4 or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Static.

5.7.35 DDRCTRL ZQ control register 1 (DDRCTRL_ZQCTL1)

Address offset: 0x184

Reset value: 0x0200 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	T_ZQ_RESET_NOP[9:0]										T_ZQ_SHORT_INTERVAL_X1024[19:16]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
T_ZQ_SHORT_INTERVAL_X1024[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:20 **T_ZQ_RESET_NOP[9:0]**: tZQReset: Number of DFI clock cycles of NOP required after a ZQReset (ZQ calibration Reset) command is issued to SDRAM.

When the controller is operating in 1:2 frequency ratio mode, program this to tZQReset/2 and round it up to the next integer value.

This is only present for designs supporting LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Static.

Bits 19:0 **T_ZQ_SHORT_INTERVAL_X1024[19:0]**: Average interval to wait between automatically issuing ZQCS (ZQ calibration short)/MPC(ZQ calibration) commands to DDR3/DDR4/LPDDR2/LPDDR3/LPDDR4 devices.

Meaningless, if ZQCTL0.dis_auto_zq=1.

Unit: 1024 DFI clock cycles.

This is only present for designs supporting DDR3/DDR4 or LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Static.

5.7.36 DDRCTRL ZQ control register 2 (DDRCTRL_ZQCTL2)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ZQ_RESET**:

Setting this register bit to 1 triggers a ZQ Reset operation. When the ZQ Reset operation is complete, the DDRCTRL automatically clears this bit. It is recommended NOT to set this register bit if in Init, in Self-Refresh(except LPDDR4) or SR-Powerdown(LPDDR4) or Deep power-down operating modes.

For Self-Refresh(except LPDDR4) or SR-Powerdown(LPDDR4) it is scheduled after SR(except LPDDR4) or SPRD(LPDDR4) has been exited.

For Deep power down, it is not scheduled, although ZQSTAT.zq_reset_busy is de-asserted.

This is only present for designs supporting LPDDR2/LPDDR3/LPDDR4 devices.

Programming mode: Dynamic

5.7.37 DDRCTRL ZQ status register (DDRCTRL_ZQSTAT)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET_BUSY
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ZQ_RESET_BUSY**: SoC core may initiate a ZQ Reset operation only if this signal is low. This signal goes high in the clock after the DDRCTRL accepts the ZQ Reset request. It goes low when the ZQ Reset command is issued to the SDRAM and the associated NOP period is over. It is recommended not to perform ZQ Reset commands when this signal is high.
- 0 - Indicates that the SoC core can initiate a ZQ Reset operation
- 1 - Indicates that ZQ Reset operation is in progress

Programming mode: Dynamic

5.7.38 DDRCTRL DFI timing register 0 (DDRCTRL_DFITMG0)

Address offset: 0x190

Reset value: 0x0702 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DFI_T_CTRL_DELAY[4:0]					Res.	DFI_T_RDDATA_EN[6:0]						
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DFI_TPHY_WRDATA[5:0]						Res.	Res.	DFI_TPHY_WRLAT[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DFI_T_CTRL_DELAY[4:0]**: Specifies the number of DFI clock cycles after an assertion or de-assertion of the DFI control signals that the control signals at the PHY-DRAM interface reflect the assertion or de-assertion. If the DFI clock and the memory clock are not phase-aligned, this timing parameter should be rounded up to the next integer value. Note that if using RDIMM/LRDIMM, it is necessary to increment this parameter by RDIMM's/LRDIMM's extra cycle of latency in terms of DFI clock.

Programming mode: Quasi-dynamic Group 4

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **DFI_T_RDDATA_EN[6:0]**: Time from the assertion of a read command on the DFI interface to the assertion of the dfi_rddata_en signal.

Refer to PHY specification for correct value.

This corresponds to the DFI parameter trddata_en. Note that, depending on the PHY, if using RDIMM/LRDIMM, it may be necessary to use the adjusted value of CL in the calculation of trddata_en. This is to compensate for the extra cycle(s) of latency through the RDIMM/LRDIMM.

Unit: DFI clock cycles or DFI PHY clock cycles, depending on DFITMG0.dfi_rddata_use_dfi_phy_clk.

Programming mode: Quasi-dynamic Group 1 and Group 4

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DFI_TPHY_WRDATA[5:0]**: Specifies the number of clock cycles between when dfi_wrdata_en is asserted to when the associated write data is driven on the dfi_wrdata signal. This corresponds to the DFI timing parameter tphy_wrdata. Refer to PHY specification for correct value. Note, max supported value is 8.

Unit: DFI clock cycles or DFI PHY clock cycles, depending on DFITMG0.dfi_wrdata_use_dfi_phy_clk.

Programming mode: Quasi-dynamic Group 4

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DFI_TPHY_WRLAT[5:0]**: Write latency

Number of clocks from the write command to write data enable (dfi_wrdata_en). This corresponds to the DFI timing parameter tphy_wrlat.

Refer to PHY specification for correct value.Note that, depending on the PHY, if using RDIMM/LRDIMM, it may be necessary to use the adjusted value of CL in the calculation of tphy_wrlat. This is to compensate for the extra cycle(s) of latency through the RDIMM/LRDIMM.

For LPDDR4, dfi_tphy_wrlat>60 is not supported.

Unit: DFI clock cycles or DFI PHY clock cycles, depending on DFITMG0.dfi_wrdata_use_dfi_phy_clk.

Programming mode: Quasi-dynamic Group 1 and Group 4

5.7.39 DDRCTRL DFI timing register 1 (DDRCTRL_DFITMG1)

Address offset: 0x194

Reset value: 0x0000 0404

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_WRDATA_DELAY[4:0]				
											rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	DFI_T_DRAM_CLK_DISABLE[4:0]					Res.	Res.	Res.	DFI_T_DRAM_CLK_ENABLE[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **DFI_T_WRDATA_DELAY[4:0]**: Specifies the number of DFI clock cycles between when the dfi_wrdata_en

signal is asserted and when the corresponding write data transfer is completed on the DRAM bus.

This corresponds to the DFI timing parameter twrdata_delay. Refer to PHY specification for correct value.

For DFI 3.0 PHY, set to twrdata_delay, a new timing parameter introduced in DFI 3.0.

For DFI 2.1 PHY, set to tphy_wrdata + (delay of DFI write data to the DRAM).

Value to be programmed is in terms of DFI clocks, not PHY clocks.

In FREQ_RATIO=2, divide PHY's value by 2 and round up to next integer.

If using DFITMG0.dfi_wrdata_use_dfi_phy_clk=1, add 1 to the value.

Unit: Clocks

Programming mode: Quasi-dynamic Group 4

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DFI_T_DRAM_CLK_DISABLE[4:0]**: Specifies the number of DFI clock cycles from the assertion of the dfi_dram_clk_disable signal on the DFI until the clock to the DRAM memory devices, at the PHY-DRAM boundary, maintains a low value. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value.

Programming mode: Quasi-dynamic Group 4

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DFI_T_DRAM_CLK_ENABLE[4:0]**: Specifies the number of DFI clock cycles from the de-assertion of the dfi_dram_clk_disable signal on the DFI until the first valid rising edge of the clock to the DRAM memory devices, at the PHY-DRAM boundary. If the DFI clock and the memory clock are not phase aligned, this timing parameter should be rounded up to the next integer value.

Programming mode: Quasi-dynamic Group 4

5.7.40 DDRCTRL low power configuration register 0 (DDRCTRL_DFILPCFG0)

Address offset: 0x198

Reset value: 0x0700 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DFI_TLP_RESP[4:0]					DFI_LP_WAKEUP_DPD[3:0]				Res.	Res.	Res.	DFI_LP_EN_DPD
			rw	rw	rw	rw	rw	rw	rw	rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DFI_LP_WAKEUP_SR[3:0]				Res.	Res.	Res.	DFI_LP_EN_SR	DFI_LP_WAKEUP_PD[3:0]				Res.	Res.	Res.	DFI_LP_EN_PD
rw	rw	rw	rw				rw	rw	rw	rw	rw				rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DFI_TLP_RESP[4:0]**: Setting in DFI clock cycles for DFI's tlp_resp time.

Same value is used for both Power Down, Self Refresh, Deep Power Down and Maximum Power Saving modes.

DFI 2.1 specification onwards, recommends using a fixed value of 7 always.

Programming mode: Static

Bits 23:20 **DFI_LP_WAKEUP_DPD[3:0]**: Value in DFI clock cycles to drive on dfi_lp_wakeup signal when Deep Power Down mode is entered.

Determines the DFI's tlp_wakeup time:

- 0x0 - 16 cycles
- 0x1 - 32 cycles
- 0x2 - 64 cycles
- 0x3 - 128 cycles
- 0x4 - 256 cycles
- 0x5 - 512 cycles
- 0x6 - 1024 cycles
- 0x7 - 2048 cycles
- 0x8 - 4096 cycles
- 0x9 - 8192 cycles
- 0xA - 16384 cycles
- 0xB - 32768 cycles
- 0xC - 65536 cycles
- 0xD - 131072 cycles
- 0xE - 262144 cycles
- 0xF - Unlimited

This is only present for designs supporting mDDR or LPDDR2/LPDDR3 devices.

Programming mode: Static.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **DFI_LP_EN_DPD**: Enables DFI Low-power interface handshaking during Deep Power Down Entry/Exit.

- 0 - Disabled
- 1 - Enabled

This is only present for designs supporting mDDR or LPDDR2/LPDDR3 devices.

Programming mode: Static.

Bits 15:12 **DFI_LP_WAKEUP_SR[3:0]**: Value in DFI clpck cycles to drive on dfi_lp_wakeup signal when Self Refresh mode is entered.

Determines the DFI's tlp_wakeup time:

- 0x0 - 16 cycles
- 0x1 - 32 cycles
- 0x2 - 64 cycles
- 0x3 - 128 cycles
- 0x4 - 256 cycles
- 0x5 - 512 cycles
- 0x6 - 1024 cycles
- 0x7 - 2048 cycles
- 0x8 - 4096 cycles
- 0x9 - 8192 cycles
- 0xA - 16384 cycles
- 0xB - 32768 cycles
- 0xC - 65536 cycles
- 0xD - 131072 cycles
- 0xE - 262144 cycles
- 0xF - Unlimited

Programming mode: Static

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **DFI_LP_EN_SR**: Enables DFI Low Power interface handshaking during Self Refresh Entry/Exit.

- 0 - Disabled
- 1 - Enabled

Programming mode: Static

Bits 7:4 **DFI_LP_WAKEUP_PD[3:0]**: Value in DFI clock cycles to drive on dfi_lp_wakeup signal when Power Down mode is entered.

Determines the DFI's tlp_wakeup time:

- 0x0 - 16 cycles
- 0x1 - 32 cycles
- 0x2 - 64 cycles
- 0x3 - 128 cycles
- 0x4 - 256 cycles
- 0x5 - 512 cycles
- 0x6 - 1024 cycles
- 0x7 - 2048 cycles
- 0x8 - 4096 cycles
- 0x9 - 8192 cycles
- 0xA - 16384 cycles
- 0xB - 32768 cycles
- 0xC - 65536 cycles
- 0xD - 131072 cycles
- 0xE - 262144 cycles
- 0xF - Unlimited

Programming mode: Static

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DFI_LP_EN_PD**: Enables DFI Low Power interface handshaking during Power Down Entry/Exit.
- 0 - Disabled
- 1 - Enabled

Programming mode: Static

5.7.41 DDRCTRL DFI update register 0 (DDRCTRL_DFIUPD0)

Address offset: 0x1A0

Reset value: 0x0040 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS_AUTO_CTRLUPD	DIS_AUTO_CTRLUPD_SRX	CTRLUPD_PRE_SRX	Res.	Res.	Res.	DFI_T_CTRLUP_MAX[9:0]									
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUP_MIN[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DIS_AUTO_CTRLUPD**: When '1', disable the automatic dfi_ctrlupd_req generation by the DDRCTRL. The core must issue the dfi_ctrlupd_req signal using register DBGCMD.ctrlupd.
When '0', DDRCTRL issues dfi_ctrlupd_req periodically.
Programming mode: Quasi-dynamic Group 3

Bit 30 **DIS_AUTO_CTRLUPD_SRX**: When '1', disable the automatic dfi_ctrlupd_req generation by the DDRCTRL at self-refresh exit.
When '0', DDRCTRL issues a dfi_ctrlupd_req before or after exiting self-refresh, depending on DFIUPD0.ctrlupd_pre_srx.
Programming mode: Static

Bit 29 **CTRLUPD_PRE_SRX**: Selects dfi_ctrlupd_req requirements at SRX:
- 0 : send ctrlupd after SRX
- 1 : send ctrlupd before SRX

If DFIUPD0.dis_auto_ctrlupd_srx=1, this register has no impact, because no dfi_ctrlupd_req is issued when SRX.
Programming mode: Static

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:16 **DFI_T_CTRLUP_MAX[9:0]**: Specifies the maximum number of DFI clock cycles that the dfi_ctrlupd_req signal can assert. Lowest value to assign to this variable is 0x40.
Programming mode: Static

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **DFI_T_CTRLUP_MIN[9:0]**: Specifies the minimum number of DFI clock cycles that the dfi_ctrlupd_req signal must be asserted. The DDRCTRL expects the PHY to respond within this time. If the PHY does not respond, the DDRCTRL de-asserts dfi_ctrlupd_req after dfi_t_ctrlup_min + 2 cycles. Lowest value to assign to this variable is 0x3.

Programming mode: Static

5.7.42 DDRCTRL DFI update register 1 (DDRCTRL_DFIUPD1)

Address offset: 0x1A4

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MIN_X1024[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MAX_X1024[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **DFI_T_CTRLUPD_INTERVAL_MIN_X1024[7:0]**: This is the minimum amount of time between DDRCTRL initiated DFI update requests (which is executed whenever the DDRCTRL is idle). Set this number higher to reduce the frequency of update requests, which can have a small impact on the latency of the first read request when the DDRCTRL is idle. Minimum allowed value for this field is 1.

Unit: 1024 DFI clock cycles.

Programming mode: Static.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DFI_T_CTRLUPD_INTERVAL_MAX_X1024[7:0]**: This is the maximum amount of time between DDRCTRL initiated DFI update requests. This timer resets with each update request; when the timer expires dfi_ctrlupd_req is sent and traffic is blocked until the dfi_ctrlupd_ackx is received. PHY can use this idle time to recalibrate the delay lines to the DLLs. The DFI controller update is also used to reset PHY FIFO pointers in case of data capture errors. Updates are required to maintain calibration over PVT, but frequent updates may impact performance. Minimum allowed value for this field is 1.

Note: Value programmed for DFIUPD1.dfi_t_ctrlupd_interval_max_x1024 must be greater than DFIUPD1.dfi_t_ctrlupd_interval_min_x1024.

Unit: 1024 DFI clock cycles.

Programming mode: Static.

5.7.43 DDRCTRL DFI update register 2 (DDRCTRL_DFIUPD2)

Address offset: 0x1A8

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFI_PHYUPD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **DFI_PHYUPD_EN**: Enables the support for acknowledging PHY-initiated updates:
- 0 - Disabled
- 1 - Enabled

Programming mode: Static

Bits 30:0 Reserved, must be kept at reset value.

5.7.44 DDRCTRL DFI miscellaneous control register (DDRCTRL_DFIMISC)

Address offset: 0x1B0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	DFI_FREQUENCY[4:0]					Res.	Res.	DFI_INIT_START	CTL_IDLE_EN	Res.	Res.	Res.	DFI_INIT_COMPLETE_EN
			rw	rw	rw	rw	rw			rw	rw				rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **DFI_FREQUENCY[4:0]**: Indicates the operating frequency of the system. The number of supported frequencies and the mapping of signal values to clock frequencies are defined by the PHY.

Programming mode: Quasi-dynamic Group 1

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **DFI_INIT_START**: PHY init start request signal.When asserted it triggers the PHY init start request
Programming mode: Quasi-dynamic Group 3

Bit 4 **CTL_IDLE_EN**: Enables support of ctl_idle signal
Programming mode: Static

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DFI_INIT_COMPLETE_EN**: PHY initialization complete enable signal.
When asserted the dfi_init_complete signal can be used to trigger SDRAM initialisation
Programming mode: Quasi-dynamic Group 3

5.7.45 DDRCTRL DFI status register (DDRCTRL_DFISTAT)

Address offset: 0x1BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_LP_ACK	DFI_INIT_COMPLETE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DFI_LP_ACK**: Stores the value of the dfi_lp_ack input to the controller.
Programming mode: Dynamic

Bit 0 **DFI_INIT_COMPLETE**: The status flag register which announces when the DFI initialization has been completed. The DFI INIT triggered by dfi_init_start signal and then the dfi_init_complete flag is polled to know when the initialization is done.
Programming mode: Dynamic

5.7.46 DDRCTRL DFI PHY master register (DDRCTRL_DFIPHYMSTR)

Address offset: 0x1C4

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_PHYMSTR_EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **DFI_PHYMSTR_EN**: Enables the PHY Master Interface:
- 0 - Disabled
- 1 - Enabled

Programming mode: Dynamic

5.7.47 DDRCTRL address map register 1 (DDRCTRL_ADDRMAP1)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_BANK_B2[5:0]					
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	ADDRMAP_BANK_B1[5:0]						Res.	Res.	ADDRMAP_BANK_B0[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **ADDRMAP_BANK_B2[5:0]**: Selects the HIF address bit used as bank address bit 2.

Valid Range: 0 to 31, and 63

Internal Base: 4

The selected HIF address bit is determined by adding the internal base to the value of this field. If unused, set to 63 and then bank address bit 2 is set to 0.

Programming mode: Static.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **ADDRMAP_BANK_B1[5:0]**: Selects the HIF address bits used as bank address bit 1.
Valid Range: 0 to 32, and 63
Internal Base: 3
The selected HIF address bit for each of the bank address bits is determined by adding the internal base to the value of this field.
If unused, set to 63 and then bank address bit 1 is set to 0.
Programming mode: Static.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **ADDRMAP_BANK_B0[5:0]**: Selects the HIF address bits used as bank address bit 0.
Valid Range: 0 to 32, and 63
Internal Base: 2
The selected HIF address bit for each of the bank address bits is determined by adding the internal base to the value of this field.
If unused, set to 63 and then bank address bit 0 is set to 0.
Programming mode: Static.

5.7.48 DDRCTRL address map register 2 (DDRCTRL_ADDRMAP2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_COL_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B4[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_COL_B3[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B2[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_COL_B5[3:0]**:
- Full bus width mode: Selects the HIF address bit used as column address bit 5.
- Half bus width mode: Selects the HIF address bit used as column address bit 6.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 7 .

Valid Range: 0 to 7, and 15
Internal Base: 5
The selected HIF address bit is determined by adding the internal base to the value of this field.
If unused, set to 15 and then this column address bit is set to 0.
Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_COL_B4[3:0]**:
- Full bus width mode: Selects the HIF address bit used as column address bit 4.
- Half bus width mode: Selects the HIF address bit used as column address bit 5.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 6.

Valid Range: 0 to 7, and 15

Internal Base: 4

The selected HIF address bit is determined by adding the internal base to the value of this field.

If unused, set to 15 and then this column address bit is set to 0.

Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_COL_B3[3:0]**:
- Full bus width mode: Selects the HIF address bit used as column address bit 3.
- Half bus width mode: Selects the HIF address bit used as column address bit 4.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 5.

Valid Range: 0 to 7

Internal Base: 3

The selected HIF address bit is determined by adding the internal base to the value of this field.

Note, if UMCTL2_INCL_ARB=1, MEMC_BURST_LENGTH=16, Full bus width (MSTR.data_bus_width=00) and BL16 (MSTR.burst_rdwr=1000), it is recommended to program this to 0.

Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_COL_B2[3:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 2.
- Half bus width mode: Selects the HIF address bit used as column address bit 3.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 4.

Valid Range: 0 to 7

Internal Base: 2

The selected HIF address bit is determined by adding the internal base to the value of this field.

Note, if UMCTL2_INCL_ARB=1 and MEMC_BURST_LENGTH=8, it is required to program this to 0 unless:

- in Half or Quarter bus width (MSTR.data_bus_width!=00) and
- PCCFG.bl_exp_mode==1 and either
- In DDR4 and ADDRMAP8.addrmap_bg_b0==0 or
- In LPDDR4 and ADDRMAP1.addrmap_bank_b0==0

If UMCTL2_INCL_ARB=1 and MEMC_BURST_LENGTH=16, it is required to program this to 0 unless:

- in Half or Quarter bus width (MSTR.data_bus_width!=00) and
- PCCFG.bl_exp_mode==1 and
- In DDR4 and ADDRMAP8.addrmap_bg_b0==0

Otherwise, if MEMC_BURST_LENGTH=8 and Full Bus Width (MSTR.data_bus_width==00), it is recommended to program this to 0 so that HIF[2] maps to column address bit 2.

If MEMC_BURST_LENGTH=16 and Full Bus Width (MSTR.data_bus_width==00), it is recommended to program this to 0 so that HIF[2] maps to column address bit 2.

If MEMC_BURST_LENGTH=16 and Half Bus Width (MSTR.data_bus_width==01), it is recommended to program this to 0 so that HIF[2] maps to column address bit 3.

Programming mode: Static

5.7.49 DDRCTRL address map register 3 (DDRCTRL_ADDRMAP3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ADDRMAP_COL_B9[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B8[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	ADDRMAP_COL_B7[4:0]					Res.	Res.	Res.	Res.	ADDRMAP_COL_B6[3:0]			
			rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **ADDRMAP_COL_B9[4:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 9.
- Half bus width mode: Selects the HIF address bit used as column address bit 11 (10 in LPDDR2/LPDDR3 mode).
- Quarter bus width mode: Selects the HIF address bit used as column address bit 13 (11 in LPDDR2/LPDDR3 mode).

Valid Range: 0 to 7, x, and 31. x indicate a valid value in inline ECC configuration.

Internal Base: 9

The selected HIF address bit is determined by adding the internal base to the value of this field.

Note: Per JEDEC DDR2/3/mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.

In LPDDR2/LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used.

In Inline ECC configuration (MEMC_INLINE_ECC=1) and ECC is enabled (ECCCFG0.ecc_mode>0), the highest 3 column address bits must map to the highest 3 valid HIF address bits.

If column bit 9 is the highest column address bit, it must map to the highest valid HIF address bit. (x = the highest valid HIF address bit - internal base)

If column bit 9 is the second highest column address bit, it must map to the second highest valid HIF address bit. (x = the highest valid HIF address bit - 1 - internal base)

If column bit 9 is the third highest column address bit, it must map to the third highest valid HIF address bit. (x = the highest valid HIF address bit - 2 - internal base)

If unused, set to 31 and then this column address bit is set to 0.

Programming mode: Static.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **ADDRMAP_COL_B8[4:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 8.
- Half bus width mode: Selects the HIF address bit used as column address bit 9.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 11 (10 in LPDDR2/LPDDR3 mode).

Valid Range: 0 to 7, x, and 31. x indicate a valid value in inline ECC configuration.

Internal Base: 8

The selected HIF address bit is determined by adding the internal base to the value of this field.

Note: Per JEDEC DDR2/3/mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.

In LPDDR2/LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used.

In Inline ECC configuration (MEMC_INLINE_ECC=1) and ECC is enabled (ECCCFG0.ecc_mode>0), the highest 3 column address bits must map to the highest 3 valid HIF address bits.

If column bit 8 is the second highest column address bit, it must map to the second highest valid HIF address bit. (x = the highest valid HIF address bit - 1 - internal base).

If column bit 8 is the third highest column address bit, it must map to the third highest valid HIF address bit. (x = the highest valid HIF address bit - 2 - internal base).

If unused, set to 31 and then this column address bit is set to 0.

Programming mode: Static.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **ADDRMAP_COL_B7[4:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 7.
- Half bus width mode: Selects the HIF address bit used as column address bit 8.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 9.

Valid Range: 0 to 7, x, and 31. x indicate a valid value in inline ECC configuration.

Internal Base: 7

The selected HIF address bit is determined by adding the internal base to the value of this field.

In Inline ECC configuration (MEMC_INLINE_ECC=1) and ECC is enabled (ECCCFG0.ecc_mode>0), the highest 3 column address bits must map to the highest 3 valid HIF address bits.

If column bit 7 is the third highest column address bit, it must map to the third highest valid HIF address bit. (x = the highest valid HIF address bit - 2 - internal base).

If unused, set to 31 and then this column address bit is set to 0.

Programming mode: Static.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_COL_B6[3:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 6.
- Half bus width mode: Selects the HIF address bit used as column address bit 7.
- Quarter bus width mode: Selects the HIF address bit used as column address bit 8.

Valid Range: 0 to 7, and 15

Internal Base: 6

The selected HIF address bit is determined by adding the internal base to the value of this field.

If unused, set to 15 and then this column address bit is set to 0.

Programming mode: Static

5.7.50 DDRCTRL address map register 4 (DDRCTRL_ADDRMAP4)

Address offset: 0x210

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ADDRMAP_COL_B11[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B10[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **ADDRMAP_COL_B11[4:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 13 (11 in LPDDR2/LPDDR3 mode).

- Half bus width mode: UNUSED. See later in this description for value you need to set to make it unused.

- Quarter bus width mode: UNUSED. See later in this description for value you need to set to make it unused.

Valid Range: 0 to 7, x, and 31. x indicate a valid value in inline ECC configuration.

Internal Base: 11

The selected HIF address bit is determined by adding the internal base to the value of this field.

Note: Per JEDEC DDR2/3/mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.

In LPDDR2/LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used.

In Inline ECC configuration (MEMC_INLINE_ECC=1) and ECC is enabled (ECCCFG0.ecc_mode>0), the highest 3 column address bits must map to the highest 3 valid HIF address bits.

If column bit 11 is the highest column address bit, it must map to the highest valid HIF address bit. (x = the highest valid HIF address bit - internal base)

If column bit 11 is the second highest column address bit, it must map to the second highest valid HIF address bit. (x = the highest valid HIF address bit - 1 - internal base)

If column bit 11 is the third highest column address bit, it must map to the third highest valid HIF address bit. (x = the highest valid HIF address bit - 2 - internal base)

If unused, set to 31 and then this column address bit is set to 0.

Programming mode: Static

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **ADDRMAP_COL_B10[4:0]**:

- Full bus width mode: Selects the HIF address bit used as column address bit 11 (10 in LPDDR2/LPDDR3 mode).
- Half bus width mode: Selects the HIF address bit used as column address bit 13 (11 in LPDDR2/LPDDR3 mode).
- Quarter bus width mode: UNUSED. See later in this description for value you need to set to make it unused.

Valid Range: 0 to 7, x, and 31. x indicate a valid value in inline ECC configuration.

Internal Base: 10

The selected HIF address bit is determined by adding the internal base to the value of this field.

Note: Per JEDEC DDR2/3/mDDR specification, column address bit 10 is reserved for indicating auto-precharge, and hence no source address bit can be mapped to column address bit 10.

In LPDDR2/LPDDR3, there is a dedicated bit for auto-precharge in the CA bus and hence column bit 10 is used.

In Inline ECC configuration (MEMC_INLINE_ECC=1) and ECC is enabled (ECCCFG0.ecc_mode>0), the highest 3 column address bits must map to the highest 3 valid HIF address bits.

If column bit 10 is the highest column address bit, it must map to the highest valid HIF address bit. (x = the highest valid HIF address bit - internal base)

If column bit 10 is the second highest column address bit, it must map to the second highest valid HIF address bit. (x = the highest valid HIF address bit - 1 - internal base)

If column bit 10 is the third highest column address bit, it must map to the third highest valid HIF address bit. (x = the highest valid HIF address bit - 2 - internal base)

If unused, set to 31 and then this column address bit is set to 0.

Programming mode: Static

5.7.51 DDRCTRL address map register 5 (DDRCTRL_ADDRMAP5)

Address offset: 0x214

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B11[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2_10[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B1[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B0[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B11[3:0]**: Selects the HIF address bit used as row address bit 11.
Valid Range: 0 to 11, and 15
Internal Base: 17
The selected HIF address bit is determined by adding the internal base to the value of this field.
If unused, set to 15 and then row address bit 11 is set to 0.
Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B2_10[3:0]**: Selects the HIF address bits used as row address bits 2 to 10.
Valid Range: 0 to 11, and 15
Internal Base: 8 (for row address bit 2), 9 (for row address bit 3), 10 (for row address bit 4) etc increasing to 16 (for row address bit 10)
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field.
When set to 15, the values of row address bits 2 to 10 are defined by registers ADDRMAP9, ADDRMAP10, ADDRMAP11.
Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B1[3:0]**: Selects the HIF address bits used as row address bit 1.
Valid Range: 0 to 11
Internal Base: 7
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field.
Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B0[3:0]**: Selects the HIF address bits used as row address bit 0.
Valid Range: 0 to 11
Internal Base: 6
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field.
Programming mode: Static

5.7.52 DDRCTRL address register 6 (DDRCTRL_ADDRMAP6)

Address offset: 0x218

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPDDR3_6GB_12GB	Res.	Res.	Res.	ADDRMAP_ROW_B15[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B14[3:0]			
rw				rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B13[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B12[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LPDDR3_6GB_12GB**:
Set this to 1 if there is an LPDDR3 SDRAM 6Gb or 12Gb device in use.
- 1 - LPDDR3 SDRAM 6Gb/12Gb device in use. Every address having row[14:13]==2'b11 is considered as invalid
- 0 - non-LPDDR3 6Gb/12Gb device in use. All addresses are valid

Present only in designs configured to support LPDDR3.
Programming mode: Static

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B15[3:0]**: Selects the HIF address bit used as row address bit 15.
Valid Range: 0 to 11, and 15
Internal Base: 21
The selected HIF address bit is determined by adding the internal base to the value of this field. If unused, set to 15 and then row address bit 15 is set to 0.
Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B14[3:0]**: Selects the HIF address bit used as row address bit 14.
Valid Range: 0 to 11, and 15
Internal Base: 20
The selected HIF address bit is determined by adding the internal base to the value of this field. If unused, set to 15 and then row address bit 14 is set to 0.
Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B13[3:0]**: Selects the HIF address bit used as row address bit 13.
Valid Range: 0 to 11, and 15
Internal Base: 19
The selected HIF address bit is determined by adding the internal base to the value of this field. If unused, set to 15 and then row address bit 13 is set to 0.
Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B12[3:0]**: Selects the HIF address bit used as row address bit 12.
Valid Range: 0 to 11, and 15
Internal Base: 18
The selected HIF address bit is determined by adding the internal base to the value of this field. If unused, set to 15 and then row address bit 12 is set to 0.
Programming mode: Static

5.7.53 DDRCTRL address map register 9 (DDRCTRL_ADDRMAP9)

Address offset: 0x224

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B4[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B3[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B5[3:0]**: Selects the HIF address bits used as row address bit 5.
Valid Range: 0 to 11
Internal Base: 11
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B4[3:0]**: Selects the HIF address bits used as row address bit 4.
Valid Range: 0 to 11
Internal Base: 10
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B3[3:0]**: Selects the HIF address bits used as row address bit 3.
Valid Range: 0 to 11
Internal Base: 9
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B2[3:0]**: Selects the HIF address bits used as row address bit 2.
Valid Range: 0 to 11
Internal Base: 8
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

5.7.54 DDRCTRL address map register 10 (DDRCTRL_ADDRMAP10)

Address offset: 0x228

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B9[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B8[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	ADDRMAP_ROW_B7[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B6[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **ADDRMAP_ROW_B9[3:0]**: Selects the HIF address bits used as row address bit 9.
Valid Range: 0 to 11
Internal Base: 15
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ADDRMAP_ROW_B8[3:0]**: Selects the HIF address bits used as row address bit 8.
Valid Range: 0 to 11
Internal Base: 14
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **ADDRMAP_ROW_B7[3:0]**: Selects the HIF address bits used as row address bit 7.
Valid Range: 0 to 11
Internal Base: 13
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B6[3:0]**: Selects the HIF address bits used as row address bit 6.
Valid Range: 0 to 11
Internal Base: 12
The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.
Programming mode: Static

5.7.55 DDRCTRL address map register 11 (DDRCTRL_ADDRMAP11)

Address offset: 0x22C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B10[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **ADDRMAP_ROW_B10[3:0]**: Selects the HIF address bits used as row address bit 10.

Valid Range: 0 to 11

Internal Base: 16

The selected HIF address bit for each of the row address bits is determined by adding the internal base to the value of this field. This register field is used only when ADDRMAP5.addrmap_row_b2_10 is set to value 15.

Programming mode: Static

5.7.56 DDRCTRL ODT configuration register (DDRCTRL_ODTCFG)

Address offset: 0x240

Reset value: 0x0400 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WR_ODT_HOLD[3:0]				Res.	Res.	Res.	WR_ODT_DELAY[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	RD_ODT_HOLD[3:0]				Res.	RD_ODT_DELAY[4:0]					Res.	Res.
				rw	rw	rw	rw		rw	rw	rw	rw	rw		

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **WR_ODT_HOLD[3:0]**:
DFI PHY clock cycles to hold ODT for a write command. The minimum supported value is 2.
Recommended values:
DDR2:
- BL8: 0x5 (DDR2-400/533/667), 0x6 (DDR2-800), 0x7 (DDR2-1066)
- BL4: 0x3 (DDR2-400/533/667), 0x4 (DDR2-800), 0x5 (DDR2-1066)

DDR3:
- BL8: 0x6

DDR4:
- BL8: 5 + WR_PREAMBLE + CRC_MODE
  WR_PREAMBLE = 1 (1tCK write preamble), 2 (2tCK write preamble)
  CRC_MODE = 0 (not CRC mode), 1 (CRC mode)

LPDDR3:
- BL8: 7 + RU(tODTon(max)/tCK)

Programming mode: Quasi-dynamic Group 1 and Group 4

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **WR_ODT_DELAY[4:0]**:
The delay, in DFI PHY clock cycles, from issuing a write command to setting ODT values associated with that command. ODT setting must remain constant for the entire time that DQS is driven by the DDRCTRL.
Recommended values:
DDR2:
- CWL + AL - 3 (DDR2-400/533/667), CWL + AL - 4 (DDR2-800), CWL + AL - 5 (DDR2-1066)
  If (CWL + AL - 3 < 0), DDRCTRL does not support ODT for write operation.

DDR3:
- 0x0

DDR4:
- DFITMG1.dfi_t_cmd_lat (to adjust for CAL mode)

LPDDR3:
- WL - 1 - RU(tODTon(max)/tCK))

Programming mode: Quasi-dynamic Group 1 and Group 4

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **RD_ODT_HOLD[3:0]**:
DFI PHY clock cycles to hold ODT for a read command. The minimum supported value is 2.
Recommended values:
DDR2:
- BL8: 0x6 (not DDR2-1066), 0x7 (DDR2-1066)
- BL4: 0x4 (not DDR2-1066), 0x5 (DDR2-1066)

DDR3:
- BL8 - 0x6

DDR4:
- BL8: 5 + RD_PREAMBLE
  RD_PREAMBLE = 1 (1tCK write preamble), 2 (2tCK write preamble)

LPDDR3:
- BL8: 5 + RU(tDQSCK(max)/tCK) - RD(tDQSCK(min)/tCK) + RU(tODTon(max)/tCK)

Programming mode: Quasi-dynamic Group 1 and Group 4

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 **RD_ODT_DELAY[4:0]**:
The delay, in DFI PHY clock cycles, from issuing a read command to setting ODT values associated with that command. ODT setting must remain constant for the entire time that DQS is driven by the DDRCTRL.
Recommended values:
DDR2:
- CL + AL - 4 (not DDR2-1066), CL + AL - 5 (DDR2-1066)
If (CL + AL - 4 < 0), DDRCTRL does not support ODT for read operation.
DDR3:
- CL - CWL
DDR4:
- CL - CWL - RD_PREAMBLE + WR_PREAMBLE + DFITMG1.dfi_t_cmd_lat (to adjust for CAL mode)
WR_PREAMBLE = 1 (1tCK write preamble), 2 (2tCK write preamble)
RD_PREAMBLE = 1 (1tCK write preamble), 2 (2tCK write preamble)
If (CL - CWL - RD_PREAMBLE + WR_PREAMBLE) < 0, DDRCTRL does not support ODT for read operation.
LPDDR3:
- RL + RD(tDQSCK(min)/tCK) - 1 - RU(tODTon(max)/tCK)
Programming mode: Quasi-dynamic Group 1 and Group 4.

Bits 1:0 Reserved, must be kept at reset value.

5.7.57 DDRCTRL ODT/Rank map register (DDRCTRL_ODTMAP)

Address offset: 0x244

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RANK0_RD_ODT	Res.	Res.	Res.	RANK0_WR_ODT
											rw				rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RANK0_RD_ODT**: Indicates which remote ODTs must be turned on during a read from rank 0.
Each rank has a remote ODT (in the SDRAM) which can be turned on by setting the appropriate bit here.
Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, etc.
For each rank, set its bit to 1 to enable its ODT.
Programming mode: Static

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RANK0_WR_ODT**: Indicates which remote ODTs must be turned on during a write to rank 0.
Each rank has a remote ODT (in the SDRAM) which can be turned on by setting the appropriate bit here.
Rank 0 is controlled by the LSB; rank 1 is controlled by bit next to the LSB, etc.
For each rank, set its bit to 1 to enable its ODT.
Programming mode: Static

5.7.58 DDRCTRL scheduler control register (DDRCTRL_SCHED)

Address offset: 0x250

Reset value: 0x0000 0805

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RDWR_IDLE_GAP[6:0]							GO2CRITICAL_HYSTERESIS[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LPR_NUM_ENTRIES[3:0]				Res.	Res.	Res.	Res.	Res.	PAGECLOSE	PREFER_WRITE	FORCE_LOW_PRI_N
				rw	rw	rw	rw						rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **RDWR_IDLE_GAP[6:0]**: When the preferred transaction store is empty for these many clock cycles, switch to the alternate transaction store if it is non-empty.
The read transaction store (both high and low priority) is the default preferred transaction store and the write transaction store is the alternative store.
When prefer write over read is set this is reversed.
0x0 is a legal value for this register. When set to 0x0, the transaction store switching happens immediately when the switching conditions become true.
FOR PERFORMANCE ONLY
Programming mode: Static

Bits 23:16 **GO2CRITICAL_HYSTERESIS[7:0]**: UNUSED
Programming mode: Static

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **LPR_NUM_ENTRIES[3:0]**: Number of entries in the low priority transaction store is this value + 1.
(MEMC_NO_OF_ENTRY - (SCHED.lpr_num_entries + 1)) is the number of entries available for the high priority transaction store.
Setting this to maximum value allocates all entries to low priority transaction store.
Setting this to 0 allocates 1 entry to low priority transaction store and the rest to high priority transaction store.
Note: In ECC configurations, the numbers of write and low priority read credits issued is one less than in the non-ECC case. One entry each is reserved in the write and low-priority read CAMs for storing the RMW requests arising out of single bit error correction RMW operation.
Programming mode: Static

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **PAGECLOSE**: If true, bank is kept open only while there are page hit transactions available in the CAM to that bank. The last read or write command in the CAM with a bank and page hit is executed with auto-precharge if SCHED1.pageclose_timer=0. Even if this register set to 1 and SCHED1.pageclose_timer is set to 0, explicit precharge (and not auto-precharge) may be issued in some cases where there is a mode switch between write and read or between LPR and HPR. The Read and Write commands that are executed as part of the ECC scrub requests are also executed without auto-precharge.

If false, the bank remains open until there is a need to close it (to open a different page, or for page timeout or refresh timeout) - also known as open page policy. The open page policy can be overridden by setting the per-command-autopre bit on the HIF interface (hif_cmd_autopre).

The pageclose feature provids a midway between Open and Close page policies.

For performance only.

Programming mode: Quasi-dynamic Group 3

Bit 1 **PREFER_WRITE**: If set then the bank selector prefers writes over reads.

FOR DEBUG ONLY.

Programming mode: Static

Bit 0 **FORCE_LOW_PRI_N**: Active low signal. When asserted ('0'), all incoming transactions are forced to low priority. This implies that all high priority read (HPR) and variable priority read commands (VPR) are treated as low priority read (LPR) commands. On the write side, all variable priority write (VPW) commands are treated as normal priority write (NPW) commands. Forcing the incoming transactions to low priority implicitly turns off bypass path for read commands.

For performance only.

Programming mode: Static

5.7.59 DDRCTRL scheduler control register 1 (DDRCTRL_SCHED1)

Address offset: 0x254

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAGECLOSE_TIMER[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PAGECLOSE_TIMER[7:0]**: This field works in conjunction with SCHED.pageclose.
It only has meaning if SCHED.pageclose==1.
If SCHED.pageclose==1 and pageclose_timer==0, then an auto-precharge may be scheduled for last read
or write command in the CAM with a bank and page hit.
Note, sometimes an explicit precharge is scheduled instead of the auto-precharge. See SCHED.pageclose for details of when this may happen.
If SCHED.pageclose==1 and pageclose_timer>0, then an auto-precharge is not scheduled for last read
or write command in the CAM with a bank and page hit.
Instead, a timer is started, with pageclose_timer as the initial value.
There is a timer on a per bank basis.
The timer decrements unless the next read or write in the CAM to a bank is a page hit.
It gets reset to pageclose_timer value if the next read or write in the CAM to a bank is a page hit.
Once the timer has reached zero, an explicit precharge is attempted to be scheduled.
Programming mode: Static.

5.7.60 DDRCTRL high priority read CAM register 1 (DDRCTRL_PERFHPR1)

Address offset: 0x25C

Reset value: 0x0F00 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPR_MAX_STARVE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **HPR_XACT_RUN_LENGTH[7:0]**:
Number of transactions that are serviced once the HPR queue goes critical is the smaller of:
- (a) This number
- (b) Number of transactions available.
Unit: Transaction.
For performance only.
Programming mode: Quasi-dynamic Group 3

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **HPR_MAX_STARVE[15:0]**: Number of DFI clocks that the HPR queue can be starved before it goes critical. The minimum valid functional value for this register is 0x1. Programming it to 0x0 disables the starvation functionality; during normal operation, this function must not be disabled as it causes excessive latencies.
For performance only.
Programming mode: Quasi-dynamic Group 3

5.7.61 DDRCTRL low priority read CAM register 1 (DDRCTRL_PERFLPR1)

Address offset: 0x264

Reset value: 0x0F00 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPR_MAX_STARVE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LPR_XACT_RUN_LENGTH[7:0]**:
Number of transactions that are serviced once the LPR queue goes critical is the smaller of:
- (a) This number
- (b) Number of transactions available.

Unit: Transaction.
For performance only
Programming mode: Quasi-dynamic group 3

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **LPR_MAX_STARVE[15:0]**: Number of DFI clocks that the LPR queue can be starved before it goes critical. The minimum valid functional value for this register is 0x1. Programming it to 0x0 disables the starvation functionality; during normal operation, this function must be disabled as it causes excessive latencies.
For performance only
Programming mode: Quasi-dynamic Group 3

5.7.62 DDRCTRL write CAM register 1 (DDRCTRL_PERFWR1)

Address offset: 0x26C

Reset value: 0x0F00 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
W_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
W_MAX_STARVE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **W_XACT_RUN_LENGTH[7:0]**:
Number of transactions that are serviced once the WR queue goes critical is the smaller of:
- (a) This number
- (b) Number of transactions available.

Unit: Transaction.
For performance only.
Programming mode: Quasi-dynamic Group 3.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **W_MAX_STARVE[15:0]**: Number of DFI clocks that the WR queue can be starved before it goes critical. The minimum valid functional value for this register is 0x1. Programming it to 0x0 disables the starvation functionality; during normal operation, this function must not be disabled as it causes excessive latencies.

For performance only.

Programming mode: Quasi-dynamic Group 3.

5.7.63 DDRCTRL debug register 0 (DDRCTRL_DBG0)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_COLLISION_PAGE_OPT	Res.	Res.	Res.	DIS_WC
											rw				rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIS_COLLISION_PAGE_OPT**: When this is set to '0', auto-precharge is disabled for the flushed command in a collision case. Collision cases are write followed by read to same address, read followed by write to same address, or write followed by write to same address with DBG0.dis_wc bit = 1 (where same address comparisons exclude the two address bits representing critical word).

FOR DEBUG ONLY.

Programming mode: Static

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DIS_WC**: When 1, disable write combine.

FOR DEBUG ONLY

Programming mode: Static

5.7.64 DDRCTRL debug register 1 (DDRCTRL_DBG1)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_HIF	DIS_DQ
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DIS_HIF**: When 1, DDRCTRL asserts the HIF command signal hif_cmd_stall. DDRCTRL ignores the hif_cmd_valid and all other associated request signals.

This bit is intended to be switched on-the-fly.

Programming mode: Dynamic

Bit 0 **DIS_DQ**: When 1, DDRCTRL does not de-queue any transactions from the CAM. Bypass is also disabled. All transactions are queued in the CAM. No reads or writes are issued to SDRAM as long as this is asserted.

This bit may be used to prevent reads or writes being issued by the DDRCTRL, which makes it safe to modify certain register fields associated with reads and writes (see User Guide for details). After setting this bit, it is strongly recommended to poll DBGCAM.wr_data_pipeline_empty and DBGCAM.rd_data_pipeline_empty, before making changes to any registers which affect reads and writes. This ensures that the relevant logic in the DDRC is idle.

This bit is intended to be switched on-the-fly.

Programming mode: Dynamic

5.7.65 DDRCTRL CAM debug register (DDRCTRL_DBGCAM)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	WR_DATA_PIPELINE_EMPTY	RD_DATA_PIPELINE_EMPTY	Res.	DBG_WR_Q_EMPTY	DBG_RD_Q_EMPTY	DBG_STALL	Res.	Res.	Res.	DBG_W_Q_DEPTH[4:0]				
		r	r		r	r	r				r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	DBG_LPR_Q_DEPTH[4:0]					Res.	Res.	Res.	DBG_HPR_Q_DEPTH[4:0]				
			r	r	r	r	r				r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **WR_DATA_PIPELINE_EMPTY**: This bit indicates that the write data pipeline on the DFI interface is empty. This register is intended to be polled at least twice after setting DBG1.dis_dq, to ensure that all remaining commands/data have completed.

Programming mode: Dynamic.

Bit 28 **RD_DATA_PIPELINE_EMPTY**: This bit indicates that the read data pipeline on the DFI interface is empty. This register is intended to be polled at least twice after setting DBG1.dis_dq, to ensure that all remaining commands/data have completed.

Programming mode: Dynamic.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **DBG_WR_Q_EMPTY**: When 1, all the Write command queues and Write data buffers inside DDRC are empty. This register is to be used for debug purpose.

An example use-case scenario: When Controller enters Self-Refresh using the Low-Power entry sequence, Controller is expected to have executed all the commands in its queues and the write and read data drained. Hence this register should be 1 at that time.

For debug only.

Programming mode: Dynamic.

Bit 25 **DBG_RD_Q_EMPTY**: When 1, all the Read command queues and Read data buffers inside DDRC are empty. This register is to be used for debug purpose.

An example use-case scenario: When Controller enters Self-Refresh using the Low-Power entry sequence, Controller is expected to have executed all the commands in its queues and the write and read data drained. Hence this register should be 1 at that time.

For debug only.

Programming mode: Dynamic

Bit 24 **DBG_STALL**: Stall

For debug only.

Programming mode: Dynamic

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DBG_W_Q_DEPTH[4:0]**: Write queue depth

The last entry of WR queue is reserved for ECC SCRUB operation. This entry is not included in the calculation of the queue depth.

For debug only.

Programming mode: Dynamic

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBG_LPR_Q_DEPTH[4:0]**: Low priority read queue depth

The last entry of Lpr queue is reserved for ECC SCRUB operation. This entry is not included in the calculation of the queue depth.

For debug only.

Programming mode: Dynamic

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBG_HPR_Q_DEPTH[4:0]**: High priority read queue depth

For debug only.

Programming mode: Dynamic

5.7.66 DDRCTRL command debug register (DDRCTRL_DBGCMD)

Address offset: 0x30C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD	ZQ_CALIB_SHORT	Res.	Res.	Res.	RANK0_REFRESH
										rs	rs				rs

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CTRLUPD**:

Setting this register bit to 1 indicates to the DDRCTRL to issue a dfi_ctrlupd_req to the PHY. When this request is stored in the DDRCTRL, the bit is automatically cleared. This operation must only be performed when DFIUPD0.dis_auto_ctrlupd=1.

Programming mode: Dynamic.

Bit 4 **ZQ_CALIB_SHORT**:

Setting this register bit to 1 indicates to the DDRCTRL to issue a ZQCS (ZQ calibration short)/MPC(ZQ calibration) command to the SDRAM. When this request is stored in the DDRCTRL, the bit is automatically cleared. This operation can be performed only when ZQCTL0.dis_auto_zq=1. It is recommended NOT to set this register bit if in Init, in Self-Refresh(except LPDDR4) or SR-Powerdown(LPDDR4) or Deep power-down operating modes or maximum power saving mode.

For self-refresh(except LPDDR4) or SR-powerdown(LPDDR4) it is scheduled after SR(except LPDDR4) or SPRD(LPDDR4) has been exited.

For Deep power down and maximum power saving mode, it is not scheduled, although DBGSTAT.zq_calib_short_busy is de-asserted.

Programming mode: Dynamic.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RANK0_REFRESH**:

Setting this register bit to 1 indicates to the DDRCTRL to issue a refresh to rank 0. Writing to this bit causes DBGSTAT.rank0_refresh_busy to be set. When DBGSTAT.rank0_refresh_busy is cleared, the command has been stored in DDRCTRL.

For 3DS configuration, refresh is sent to rank index 0.

This operation can be performed only when RFSHCTL3.dis_auto_refresh=1. It is recommended NOT to set this register bit if in Init or Deep power-down operating modes or Maximum Power Saving Mode.

Programming mode: Dynamic.

5.7.67 DDRCTRL status debug register (DDRCTRL_DBGSTAT)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD_BUSY	ZQ_CALIB_SHORT_BUSY	Res.	Res.	Res.	RANK0_REFRESH_BUSY
										r	r				r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CTRLUPD_BUSY**: SoC core may initiate a ctrlupd operation only if this signal is low. This signal goes high in the clock after the DDRCTRL accepts the ctrlupd request. It goes low when the ctrlupd operation is initiated in the DDRCTRL. It is recommended not to perform ctrlupd operations when this signal is high.

- 0 - Indicates that the SoC core can initiate a ctrlupd operation
- 1 - Indicates that ctrlupd operation has not been initiated yet in the DDRCTRL

Programming mode: Dynamic

Bit 4 **ZQ_CALIB_SHORT_BUSY**: SoC core may initiate a ZQCS (ZQ calibration short) operation only if this signal is low. This signal goes high in the clock after the DDRCTRL accepts the ZQCS request. It goes low when the ZQCS operation is initiated in the DDRCTRL. It is recommended not to perform ZQCS operations when this signal is high.

- 0 - Indicates that the SoC core can initiate a ZQCS operation
- 1 - Indicates that ZQCS operation has not been initiated yet in the DDRCTRL

Programming mode: Dynamic

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RANK0_REFRESH_BUSY**: SoC core may initiate a rank0_refresh operation (refresh operation to rank 0) only if this signal is low. This signal goes high in the clock after DBGCMD.rank0_refresh is set to one. It goes low when the rank0_refresh operation is stored in the DDRCTRL. It is recommended not to perform rank0_refresh operations when this signal is high.

- 0 - Indicates that the SoC core can initiate a rank0_refresh operation
- 1 - Indicates that rank0_refresh operation has not been stored yet in the DDRCTRL

Programming mode: Dynamic

5.7.68 DDRCTRL software register programming control enable (DDRCTRL_SWCTL)

Address offset: 0x320

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SW_DONE**: Enable quasi-dynamic register programming outside reset. Program register to 0 to enable quasi-dynamic programming. Set back register to 1 once programming is done.

Programming mode: Dynamic

5.7.69 DDRCTRL software register programming control status (DDRCTRL_SWSTAT)

Address offset: 0x324

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE_ACK
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SW_DONE_ACK**: Register programming done. This register is the echo of SWCTL.sw_done. Wait for sw_done value 1 to propagate to sw_done_ack at the end of the programming sequence to ensure that the correct registers values are propagated to the destination clock domains.

Programming mode: Static

5.7.70 DDRCTRL AXI Poison configuration register (DDRCTRL_POISONCFG)

Address offset: 0x36C

Reset value: 0x0011 0011

AXI Poison configuration register common for all AXI ports.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_CLR	Res.	Res.	Res.	RD_POISON_INTR_EN	Res.	Res.	Res.	RD_POISON_SLVERR_EN
							rc_w1				rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_CLR	Res.	Res.	Res.	WR_POISON_INTR_EN	Res.	Res.	Res.	WR_POISON_SLVERR_EN
							rc_w1				rw				rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RD_POISON_INTR_CLR**: Interrupt clear for read transaction poisoning. Allow 2/3 clock cycles for correct value to propagate to core logic and clear the interrupts.
Programming mode: Dynamic

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **RD_POISON_INTR_EN**: If set to 1, enables interrupts for read transaction poisoning
Programming mode: Dynamic

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RD_POISON_SLVERR_EN**: If set to 1, enables SLVERR response for read transaction poisoning
Programming mode: Dynamic

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **WR_POISON_INTR_CLR**: Interrupt clear for write transaction poisoning. Allow 2/3 clock cycles for correct value to propagate to core logic and clear the interrupts.
Programming mode: Dynamic

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WR_POISON_INTR_EN**: If set to 1, enables interrupts for write transaction poisoning
Programming mode: Dynamic

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **WR_POISON_SLVERR_EN**: If set to 1, enables SLVERR response for write transaction poisoning
Programming mode: Dynamic

5.7.71 DDRCTRL AXI Poison status register (DDRCTRL_POISONSTAT)

Address offset: 0x370

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_1	RD_POISON_INTR_0
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_1	WR_POISON_INTR_0
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **RD_POISON_INTR_1**: Read transaction poisoning error interrupt for port 1. This register is a APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port's read address channel. Bit 0 corresponds to Port 0, and so on. Interrupt is cleared by register rd_poison_intr_clr, then value propagated to APB clock.
Programming mode: Dynamic

Bit 16 **RD_POISON_INTR_0**: Read transaction poisoning error interrupt for port 0. This register is a APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port's read address channel. Bit 0 corresponds to Port 0, and so on. Interrupt is cleared by register rd_poison_intr_clr, then value propagated to APB clock.
Programming mode: Dynamic

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **WR_POISON_INTR_1**: Write transaction poisoning error interrupt for port 1. This register is a APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port's write address channel. Bit 0 corresponds to Port 0, and so on. Interrupt is cleared by register wr_poison_intr_clr, then value propagated to APB clock.
Programming mode: Dynamic

Bit 0 **WR_POISON_INTR_0**: Write transaction poisoning error interrupt for port 0. This register is a APB clock copy (double register synchronizer) of the interrupt asserted when a transaction is poisoned on the corresponding AXI port's write address channel. Bit 0 corresponds to Port 0, and so on. Interrupt is cleared by register wr_poison_intr_clr, then value propagated to APB clock.
Programming mode: Dynamic

5.7.72 DDRCTRL port status register (DDRCTRL_PSTAT)

Address offset: 0x3FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_BUSY_1	WR_PORT_BUSY_0
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_PORT_BUSY_1	RD_PORT_BUSY_0
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **WR_PORT_BUSY_1**: Indicates if there are outstanding writes for AXI port 1.
Programming mode: Dynamic

Bit 16 **WR_PORT_BUSY_0**: Indicates if there are outstanding writes for AXI port 0.
Programming mode: Dynamic

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **RD_PORT_BUSY_1**: Indicates if there are outstanding reads for AXI port 1.
Programming mode: Dynamic

Bit 0 **RD_PORT_BUSY_0**: Indicates if there are outstanding reads for AXI port 0.
Programming mode: Dynamic

5.7.73 DDRCTRL port common configuration register (DDRCTRL_PCCFG)

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL_EXP_MODE	Res.	Res.	Res.	PAGEMATCH_LIMIT	Res.	Res.	Res.	GO2CRITICAL_EN
							rw				rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BL_EXP_MODE**: Burst length expansion mode. By default (i.e. bl_exp_mode==0) XPI expands every AXI burst into multiple HIF commands, using the memory burst length as a unit. If set to 1, then XPI uses half of the memory burst length as a unit.

This applies to both reads and writes. When MSTR.data_bus_width==00, setting bl_exp_mode to 1 has no effect.

This can be used in cases where Partial Writes is enabled (UMCTL2_PARTIAL_WR=1), in order to avoid or minimize t_ccd_l penalty in DDR4 and t_ccd_mw penalty in LPDDR4. Hence, bl_exp_mode=1 is only recommended if DDR4 or LPDDR4.

Note that if DBICTL.dm_en=0, functionality is not supported in the following cases:

- UMCTL2_PARTIAL_WR=0
- UMCTL2_PARTIAL_WR=1, MSTR.data_bus_width=01, MEMC_BURST_LENGTH=8 and MSTR.burst_rdwr=1000 (LPDDR4 only)
- UMCTL2_PARTIAL_WR=1, MSTR.data_bus_width=01, MEMC_BURST_LENGTH=4 and MSTR.burst_rdwr=0100 (DDR4 only), with either MSTR.burstchop=0 or CRCPARCTL1.crc_enable=1

Functionality is also not supported if Data Channel Interleave is enabled.

Programming mode: Static.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PAGEMATCH_LIMIT**: Page match four limit. If set to 1, limits the number of consecutive same page DDRC transactions that can be granted by the Port Arbiter to four when Page Match feature is enabled. If set to 0, there is no limit imposed on number of consecutive same page DDRC transactions.

Programming mode: Static.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GO2CRITICAL_EN**: If set to 1 (enabled), sets co_gs_go2critical_wr and co_gs_go2critical_lpr/co_gs_go2critical_hpr signals going to DDRC based on urgent input (awurgent, arurgent) coming from AXI master. If set to 0 (disabled), co_gs_go2critical_wr and co_gs_go2critical_lpr/co_gs_go2critical_hpr signals at DDRC are driven to 1b'0.
Programming mode: Static.

5.7.74 DDRCTRL port x configuration read register (DDRCTRL_PCFGR_x)

Address offset: 0x404 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDWR_ORDERED_EN
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RD_PORT_PAGEMATCH_EN	RD_PORT_URGENT_EN	RD_PORT_AGING_EN	Res.	Res.	RD_PORT_PRIORITY[9:0]									
	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RDWR_ORDERED_EN**: Enables ordered read/writes. If set to 1, preserves the ordering between read transaction and write transaction issued to the same address, on a given port. In other words, the controller ensures that all same address read and write commands from the application port interface are transported to the DFI interface in the order of acceptance. This feature is useful in cases where software coherency is desired for masters issuing back-to-back read/write transactions without waiting for write/read responses. Note that this register has an effect only if necessary logic is instantiated via the UMCTL2_RDWR_ORDERED_n parameter.
Programming mode: Static.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RD_PORT_PAGEMATCH_EN**: If set to 1, enables the Page Match feature. If enabled, once a requesting port is granted, the port is continued to be granted if the following immediate commands are to the same memory page (same bank and same row). See also related PCCFG.pagematch_limit register.
Programming mode: Static

Bit 13 **RD_PORT_URGENT_EN**: If set to 1, enables the AXI urgent sideband signal (arurgent). When enabled and arurgent is asserted by the master, that port becomes the highest priority and co_gs_go2critical_lpr/co_gs_go2critical_hpr signal to DDRC is asserted if enabled in PCCFG.go2critical_en register. Note that arurgent signal can be asserted anytime and as long as required which is independent of address handshaking (it is not associated with any particular command).

Programming mode: Static

Bit 12 **RD_PORT_AGING_EN**: If set to 1, enables aging function for the read channel of the port.

Programming mode: Static

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **RD_PORT_PRIORITY[9:0]**: Determines the initial load value of read aging counters. These counters are parallel loaded after reset, or after each grant to the corresponding port. The aging counters down-count every clock cycle where the port is requesting but not granted. The higher significant 5-bits of the read aging counter sets the priority of the read channel of a given port. Port's priority increases as the higher significant 5-bits of the counter starts to decrease. When the aging counter becomes 0, the corresponding port channel has the highest priority level (timeout condition - Priority0). For multi-port configurations, the aging counters cannot be used to set port priorities when external dynamic priority inputs (arqos) are enabled (timeout is still applicable). For single port configurations, the aging counters are only used when they timeout (become 0) to force read-write direction switching. In this case, external dynamic priority input, arqos (for reads only) can still be used to set the DDRC read priority (2 priority levels: low priority read - LPR, high priority read - HPR) on a command by command basis.

Note: The two LSBs of this register field are tied internally to 2'b00.

Programming mode: Static

5.7.75 DDRCTRL port x configuration write register (DDRCTRL_PCFGW_x)

Address offset: 0x408 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WR_PORT_PAGEMATCH_EN	WR_PORT_URGENT_EN	WR_PORT_AGING_EN	Res.	Res.	WR_PORT_PRIORITY[9:0]									
	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **WR_PORT_PAGEMATCH_EN**: If set to 1, enables the Page Match feature. If enabled, once a requesting port is granted, the port is continued to be granted if the following immediate commands are to the same memory page (same bank and same row). See also related PCCFG.pagematch_limit register.
Programming mode: Static

Bit 13 **WR_PORT_URGENT_EN**: If set to 1, enables the AXI urgent sideband signal (awurgent). When enabled and awurgent is asserted by the master, that port becomes the highest priority and co_gs_go2critical_wr signal to DDRC is asserted if enabled in PCCFG.go2critical_en register.
Note that awurgent signal can be asserted anytime and as long as required which is independent of address handshaking (it is not associated with any particular command).
Programming mode: Static

Bit 12 **WR_PORT_AGING_EN**: If set to 1, enables aging function for the write channel of the port.
Programming mode: Static

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **WR_PORT_PRIORITY[9:0]**: Determines the initial load value of write aging counters. These counters are parallel loaded after reset, or after each grant to the corresponding port. The aging counters down-count every clock cycle where the port is requesting but not granted. The higher significant 5-bits of the write aging counter sets the initial priority of the write channel of a given port. Port's priority increases as the higher significant 5-bits of the counter starts to decrease. When the aging counter becomes 0, the corresponding port channel has the highest priority level.
For multi-port configurations, the aging counters cannot be used to set port priorities when external dynamic priority inputs (awqos) are enabled (timeout is still applicable).
For single port configurations, the aging counters are only used when they timeout (become 0) to force read-write direction switching.
Note: The two LSBs of this register field are tied internally to 2'b00.
Programming mode: Static

5.7.76 DDRCTRL port x control register (DDRCTRL_PCTRL_x)

Address offset: 0x490 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **PORT_EN**: Enables AXI port n.
Programming mode: Dynamic

5.7.77 DDRCTRL port x read Q0S configuration register 0 (DDRCTRL_PCFGQOS0_x)

Address offset: 0x494 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0200 0E00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_REGION2[1:0]		Res.	Res.	RQOS_MAP_REGION1[1:0]		Res.	Res.	RQOS_MAP_REGION0[1:0]	
						rw	rw			rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL1[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **RQOS_MAP_REGION2[1:0]**: This bitfield indicates the traffic class of region2.
For dual address queue configurations, region2 maps to the red address queue.
Valid values are 1: VPR and 2: HPR only.
When VPR support is disabled (UMCTL2_VPR_EN = 0) and traffic class of region2 is set to 1 (VPR), VPR traffic is aliased to LPR traffic.
Programming mode: Quasi-dynamic Group 3

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **RQOS_MAP_REGION1[1:0]**: This bitfield indicates the traffic class of region 1.
Valid values are:
- 0 : LPR,
- 1: VPR,
- 2: HPR.

For dual address queue configurations, region1 maps to the blue address queue.
In this case, valid values are
- 0: LPR and
- 1: VPR only.

When VPR support is disabled (UMCTL2_VPR_EN = 0) and traffic class of region 1 is set to 1 (VPR), VPR traffic is aliased to LPR traffic.
Programming mode: Quasi-dynamic Group 3

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **RQOS_MAP_REGION0[1:0]**: This bitfield indicates the traffic class of region 0.
Valid values are:
- 0: LPR, 1: VPR, 2: HPR.
For dual address queue configurations, region 0 maps to the blue address queue.
In this case, valid values are:
- 0: LPR and 1: VPR only.
When VPR support is disabled (UMCTL2_VPR_EN = 0) and traffic class of region0 is set to 1 (VPR), VPR traffic is aliased to LPR traffic.
Programming mode: Quasi-dynamic Group 3

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **RQOS_MAP_LEVEL2[3:0]**: Separation level2 indicating the end of region1 mapping; start of region1 is (level1 + 1). Possible values for level2 are (level1 + 1) to 14 which corresponds to arqos.
Region2 starts from (level2 + 1) up to 15.
Note that for PA, arqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.
All of the map_level* registers must be set to distinct values.
Programming mode: Quasi-dynamic Group 3

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **RQOS_MAP_LEVEL1[3:0]**: Separation level1 indicating the end of region0 mapping; start of region0 is 0. Possible values for level1 are 0 to 13 (for dual RAQ) or 0 to 14 (for single RAQ) which corresponds to arqos.
Note that for PA, arqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.
All of the map_level* registers must be set to distinct values.
Programming mode: Quasi-dynamic Group 3

5.7.78 DDRCTRL port x read Q0S configuration register 1 (DDRCTRL_PCFGQOS1_x)

Address offset: 0x498 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RQOS_MAP_TIMEOUTR[10:0]**: Specifies the timeout value for transactions mapped to the red address queue. With single read address queue, there is no red queue and this value has no effect.
Programming mode: Quasi-dynamic Group 3

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RQOS_MAP_TIMEOUTB[10:0]**: Specifies the timeout value for transactions mapped to the blue address queue.
Programming mode: Quasi-dynamic Group 3.

5.7.79 DDRCTRL port x write Q0S configuration register 0 (DDRCTRL_PCFGWQOS0_x)

Address offset: 0x49C + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0E00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_REGION2[1:0]		Res.	Res.	WQOS_MAP_REGION1[1:0]		Res.	Res.	WQOS_MAP_REGION0[1:0]	
						rw	rw			rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL1[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **WQOS_MAP_REGION2[1:0]**: This bitfield indicates the traffic class of region 2.
Valid values are:
0: NPW, 1: VPW.
When VPW support is disabled (UMCTL2_VPW_EN = 0) and traffic class of region 2 is set to 1 (VPW), VPW traffic is aliased to NPW traffic.
Programming mode: Quasi-dynamic Group 3

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **WQOS_MAP_REGION1[1:0]**: This bitfield indicates the traffic class of region 1.
Valid values are:
0: NPW, 1: VPW.
When VPW support is disabled (UMCTL2_VPW_EN = 0) and traffic class of region 1 is set to 1 (VPW), VPW traffic is aliased to NPW traffic.
Programming mode: Quasi-dynamic Group 3

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **WQOS_MAP_REGION0[1:0]**: This bitfield indicates the traffic class of region 0.
Valid values are:
0: NPW, 1: VPW.
When VPW support is disabled (UMCTL2_VPW_EN = 0) and traffic class of region 0 is set to 1 (VPW), VPW traffic is aliased to NPW traffic.
Programming mode: Quasi-dynamic Group 3

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **WQOS_MAP_LEVEL2[3:0]**: Separation level2 indicating the end of region1 mapping; start of region1 is (level1 + 1). Possible values for level2 are (level1 + 1) to 14 which corresponds to awqos.

Region2 starts from (level2 + 1) up to 15.

Note that for PA, awqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.

All of the map_level* registers must be set to distinct values.

Programming mode: Quasi-dynamic Group 3

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WQOS_MAP_LEVEL1[3:0]**: Separation level indicating the end of region0 mapping; start of region0 is 0. Possible values for level1 are 0 to 13 which corresponds to awqos.

Note that for PA, awqos values are used directly as port priorities, where the higher the value corresponds to higher port priority.

All of the map_level* registers must be set to distinct values.

Programming mode: Quasi-dynamic Group 3

5.7.80 DDRCTRL port x write Q0S configuration register 1 (DDRCTRL_PCFGWQOS1_x)

Address offset: 0x4A0 + 0xB0 * x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT2[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT1[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **WQOS_MAP_TIMEOUT2[10:0]**: Specifies the timeout value for write transactions in region 2.

Programming mode: Quasi-dynamic Group 3.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **WQOS_MAP_TIMEOUT1[10:0]**: Specifies the timeout value for write transactions in region 0 and 1.

Programming mode: Quasi-dynamic Group 3.

5.7.81 DDRCTRL registers summary

Table 19. DDRCTRL register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DDRCTRL_MSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BURST_RDWR[3:0]				DLL_OFF_MODE	Res.	DATA_BUS_WIDTH[1:0]		Res.	EN_2T_TIMING_MODE	BURSTCHOP	Res.	Res.	Res.	Res.	Res.	LPDDR3	LPDDR2	Res.	DDR3
	Reset value													0	1	0	0	0		0	0		0	0						0	0		1
0x004	DDRCTRL_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_CAM_NOT_EMPTY	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_TYPE[1:0]		Res.	OPERATING_MODE[2:0]		
	Reset value																				0							0	0		0	0	0
0x008 - 0x00C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x010	DDRCTRL_MRCTRL0	MR_WR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_ADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_TYPE
	Reset value	0																0	0	0	0												0
0x014	DDRCTRL_MRCTRL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	DDRCTRL_MRSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MR_WR_BUSY
	Reset value																																0
0x01C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x020	DDRCTRL_DERATEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERATE_BYTE[3:0]				Res.	DERATE_VALUE[1:0]		DERATE_ENABLE
	Reset value																									0	0	0	0		0	0	0
0x024	DDRCTRL_DERATEINT	MR4_READ_INTERVAL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028 - 0x02C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x030	DDRCTRL_PWRCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_CAM_DRAIN_SELFREF	Res.	SELFREF_SW	Res.	EN_DFI_DRAM_CLK_DISABLE	DEEPPOWERDOWN_EN	POWERDOWN_EN	SELFREF_EN
	Reset value																									0		0		0	0	0	0
0x034	DDRCTRL_PWRTMG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELFREF_TO_X32[7:0]								T_DPD_X4096[7:0]								Res.	Res.	Res.	POWERDOWN_TO_X32[4:0]				
	Reset value									0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0				1	0	0	0	0
0x038	DDRCTRL_HWLPCTL	Res.	Res.	Res.	Res.	HW_LP_IDLE_X32[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LP_EXIT_IDLE_EN	HW_LP_EN
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0															1	1
0x03C - 0x04C	Res.																																
0x050	DDRCTRL_RFSHCTL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REFRESH_MARGIN[3:0]				Res.	Res.	Res.	REFRESH_TO_X32[4:0]					Res.	Res.	Res.	REFRESH_BURST[4:0]					Res.	PER_BANK_REFRESH	Res.	Res.
	Reset value									0	0	1	0				1	0	0	0	0				0	0	0	0	0		0		
0x054 - 0x05C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	DDRCTRL_RFSHCTL3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REFRESH_UPDATE_LEVEL	DIS_AUTO_REFRESH
	Reset value																															0	0
0x064	DDRCTRL_RFSHTMG	T_RFC_NOM_X1_SEL	Res.	Res.	Res.	T_RFC_NOM_X1_X32[11:0]												LPDDR3_TREFBW_EN	Res.	Res.	Res.	Res.	Res.	T_RFC_MIN[9:0]									
	Reset value	0				0	0	0	0	0	1	1	0	0	0	1	0	0						0	0	1	0	0	0	1	1	0	0
0x068 - 0x0BC	Res.																																
0x0C0	DDRCTRL_CRCPARCTL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ALERT_ERR_CNT_CLR	DFI_ALERT_ERR_INT_CLR	DFI_ALERT_ERR_INT_EN
	Reset value																														0	0	0
0x0C4 - 0x0C8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0CC	DDRCTRL_CRCPARSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_ALERT_ERR_INT	DFI_ALERT_ERR_CNT[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D0	DDRCTRL_INIT0	SKIP_DRAM_INIT[1:0]		Res.	Res.	Res.	Res.	POST_CKE_X1024[9:0]										Res.	Res.	Res.	Res.	PRE_CKE_X1024[11:0]											
	Reset value	0	0					0	0	0	0	0	0	0	0	1	0					0	0	0	0	0	1	0	0	1	1	1	0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D4	DDRCTRL_INIT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAM_RSTN_X1024[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE_OCD_X32[3:0]			
	Reset value								0	0	0	0	0	0	0	0	0													0	0	0	0
0x0D8	DDRCTRL_INIT2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLE_AFTER_RESET_X32[7:0]								Res.	Res.	Res.	Res.	MIN_STABLE_CLOCK_X1[3:0]			
	Reset value																	0	0	0	0	1	1	0	1					0	1	0	1
0x0DC	DDRCTRL_INIT3	MR[15:0]																EMR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
		0																0x0510															
0x0E0	DDRCTRL_INIT4	EMR2[15:0]																EMR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4	DDRCTRL_INIT5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ZQINIT_X32[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	MAX_AUTO_INIT_X1024[9:0]									
	Reset value									0	0	0	1	0	0	0	0							0	0	0	0	0	0	0	1	0	0
0x0E8 - 0x0EC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0F0	DDRCTRL_DIMMCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIMM_ADDR_MIRR_EN	DIMM_STAGGER_CS_EN
	Reset value																															0	0
0x0F4 - 0x0FC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	DDRCTRL_DRAMTMG0	Res.	WR2PRE[6:0]							Res.	Res.	T_FAW[5:0]						Res.	T_RAS_MAX[6:0]							Res.	Res.	T_RAS_MIN[5:0]					
	Reset value		0	0	0	1	1	1	1			0	1	0	0	0	0		0	0	1	1	0	1	1			0	0	1	1	1	1
0x104	DDRCTRL_DRAMTMG1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XP[4:0]					Res.	Res.	RD2PRE[5:0]						Res.	T_RC[6:0]						
	Reset value												0	1	0	0	0			0	0	0	1	0	0		0	0	1	0	1	0	0
0x108	DDRCTRL_DRAMTMG2	Res.	Res.	WRITE_LATENCY[5:0]						Res.	Res.	READ_LATENCY[5:0]						Res.	Res.	RD2WR[5:0]						Res.	Res.	WR2RD[5:0]					
	Reset value			0	0	0	0	1	1			0	0	0	1	0	1			0	0	0	1	1	0			0	0	1	1	0	1
0x10C	DDRCTRL_DRAMTMG3	Res.	Res.	T_MRW[9:0]										Res.	Res.	T_MRD[5:0]						Res.	Res.	T_MOD[9:0]									
	Reset value			0	0	0	0	0	0	0	1	0	1			0	0	0	1	0	0			0	0	0	0	0	0	1	1	0	0
0x110	DDRCTRL_DRAMTMG4	Res.	Res.	Res.	T_RCD[4:0]					Res.	Res.	Res.	Res.	T_CCD[3:0]				Res.	Res.	Res.	Res.	T_RRD[3:0]				Res.	Res.	Res.	T_RP[4:0]				
	Reset value				0	0	1	0	1					0	1	0	0					0	1	0	0				0	0	1	0	1
0x114	DDRCTRL_DRAMTMG5	Res.	Res.	Res.	Res.	T_CKSRX[3:0]				Res.	Res.	Res.	Res.	T_CKSRE[3:0]				Res.	Res.	T_CKESR[5:0]						Res.	Res.	Res.	T_CKE[4:0]				
	Reset value					0	1	0	1					0	1	0	1			0	0	0	1	0	0				0	0	0	1	1
0x118	DDRCTRL_DRAMTMG6	Res.	Res.	Res.	Res.	T_CKDPDE[3:0]				Res.	Res.	Res.	Res.	T_CKDPDX[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CKCSX[3:0]			
	Reset value					0	0	1	0					0	0	1	0													0	1	0	1
0x11C	DDRCTRL_DRAMTMG7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_CKPDE[3:0]				Res.	Res.	Res.	Res.	T_CKPDX[3:0]			
	Reset value																					0	0	1	0					0	0	1	0
0x120	DDRCTRL_DRAMTMG8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XS_DLL_X32[6:0]							Res.	T_XS_X32[6:0]						
	Reset value																		1	0	0	0	1	0	0		0	0	0	0	1	0	1
0x124 - 0x134	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x138	DDRCTRL_DRAMTMG14	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_XSR[11:0]											
	Reset value																					0	0	0	0	1	0	1	0	0	0	0	0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x13C	DDRCTRL_DRAMTMG15	EN_DFI_LP_T_STAB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	T_STAB_X32[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0x140 - 0x17C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x180	DDRCTRL_ZQCTL0	DIS_AUTO_ZQ	DIS_SRX_ZQCL	ZQ_RESISTOR_SHARED	Res.	Res.	T_ZQ_LONG_NOP[10:0]											Res.	Res.	Res.	Res.	Res.	Res.	T_ZQ_SHORT_NOP[9:0]									
	Reset value	0	0	0			0	1	0	0	0	0	0	0	0	0	0							0	0	0	1	0	0	0	0	0	0
0x184	DDRCTRL_ZQCTL1	Res.	Res.	T_ZQ_RESET_NOP[9:0]										T_ZQ_SHORT_INTERVAL_X1024[19:0]																			
	Reset value			0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x188	DDRCTRL_ZQCTL2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET
	Reset value																																0
0x18C	DDRCTRL_ZQSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZQ_RESET_BUSY
	Reset value																																0
0x190	DDRCTRL_DFITMG0	Res.	Res.	Res.	DFI_T_CTRL_DELAY[4:0]					Res.	DFI_T_RDDATA_EN[6:0]							Res.	Res.	DFI_TPHY_WRDATA[5:0]						Res.	Res.	DFI_TPHY_WRLAT[5:0]					
	Reset value				0	0	1	1	1		0	0	0	0	0	1	0			0	0	0	0	0	0			0	0	0	0	1	0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x194	DDRCTRL_DFITMG1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_WRDATA_DELAY[4:0]					Res.	Res.	Res.	DFI_T_DRAM_CLK_DISABLE[4:0]					Res.	Res.	Res.	DFI_T_DRAM_CLK_ENABLE[4:0]				
	Reset value												0	0	0	0	0				0	0	1	0	0				0	0	1	0	0
0x198	DDRCTRL_DFILPCFG0	Res.	Res.	Res.	DFI_TLP_RESP[4:0]					DFI_LP_WAKEUP_DPD[3:0]				Res.	Res.	Res.	DFI_LP_EN_DPD	DFI_LP_WAKEUP_SR[3:0]				Res.	Res.	Res.	DFI_LP_EN_SR	DFI_LP_WAKEUP_PD[3:0]				Res.	Res.	Res.	DFI_LP_EN_PD
	Reset value				0	0	1	1	1	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0
0x19C	Res.																																
0x1A0	DDRCTRL_DFIUPD0	DIS_AUTO_CTRLUPD	DIS_AUTO_CTRLUPD_SRX	CTRLUPD_PRE_SRX	Res.	Res.	Res.	DFI_T_CTRLUP_MAX[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUP_MIN[9:0]									
	Reset value	0	0	0				0	0	0	1	0	0	0	0	0	0							0	0	0	0	0	0	0	0	1	1
0x1A4	DDRCTRL_DFIUPD1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MIN_X1024[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_T_CTRLUPD_INTERVAL_MAX_X1024[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1A8	DDRCTRL_DFIUPD2	DFI_PHYUPD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1																															
0x1AC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x1B0	DDRCTRL_DFIMISC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_FREQUENCY[4:0]					Res.		DFI_INIT_START	CTL_IDLE_EN	Res.			DFI_INIT_COMPLETE_EN
	Reset value																				0	0	0	0	0			0	0				1
0x1B4 - 0x1B8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x1BC	DDRCTRL_DFISTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_LP_ACK	DFI_INIT_COMPLETE
	Reset value																															0	0
0x1C0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x1C4	DDRCTRL_DFIPHYMSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFI_PHYMSTR_EN
	Reset value																																1
0x1C8 - 0x200	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x204	DDRCTRL_ADDRMAP1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_BANK_B2[5:0]						Res.	Res.	ADDRMAP_BANK_B1[5:0]						Res.	Res.	ADDRMAP_BANK_B0[5:0]					
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x208	DDRCTRL_ADDRMAP2	Res.	Res.	Res.	Res.	ADDRMAP_COL_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B4[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B3[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_COL_B2[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x20C	DDRCTRL_ADDRMAP3	Res.	Res.	Res.	ADDRMAP_COL_B9[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B8[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B7[4:0]					Res.	Res.	Res.	Res.	ADDRMAP_COL_B6[3:0]			
	Reset value				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0					0	0	0	0
0x210	DDRCTRL_ADDRMAP4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_COL_B11[4:0]					Res.	Res.	Res.	ADDRMAP_COL_B10[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x214	DDRCTRL_ADDRMAP5	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B11[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2_10[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B1[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B0[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x218	DDRCTRL_ADDRMAP6	LPDDR3_6GB_12GB	Res.	Res.	Res.	ADDRMAP_ROW_B15[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B14[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B13[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B12[3:0]			
	Reset value	0				0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x21C - 0x220	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x224	DDRCTRL_ADDRMAP9	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B5[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B4[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B3[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B2[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x228	DDRCTRL_ADDRMAP10	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B9[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B8[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B7[3:0]				Res.	Res.	Res.	Res.	ADDRMAP_ROW_B6[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x22C	DDRCTRL_ADDRMAP11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRMAP_ROW_B10[3:0]			
	Reset value																													0	0	0	0
0x230 - 0x23C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x240	DDRCTRL_ODTCFG	Res.	Res.	Res.	Res.	WR_ODT_HOLD[3:0]				Res.	Res.	Res.	WR_ODT_DELAY[4:0]					Res.	Res.	Res.	Res.	RD_ODT_HOLD[3:0]				Res.	RD_ODT_DELAY[4:0]					Res.	Res.
	Reset value					0	1	0	0				0	0	0	0	0					0	1	0	0		0	0	0	0	0		
0x244	DDRCTRL_ODTMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RANK0_RD_ODT	Res.	Res.	Res.	RANK0_WR_ODT
	Reset value																												1				1
0x248 - 0x24C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x250	DDRCTRL_SCHED	Res.	RDWR_IDLE_GAP[6:0]							GO2CRITICAL_HYSTERESIS[7:0]								Res.	Res.	Res.	Res.	LPR_NUM_ENTRIES[3:0]				Res.	Res.	Res.	Res.	Res.	PAGECLOSE	PREFER_WRITE	FORCE_LOW_PRI_N
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					1	0	0	0						1	0	1
0x254	DDRCTRL_SCHED1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAGECLOSE_TIMER[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x258	Res.																																
0x25C	DDRCTRL_PERFHPR1	HPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPR_MAX_STARVE[15:0]															
	Reset value	0	0	0	0	1	1	1	1									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x260	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x264	DDRCTRL_PERFLPR1	LPR_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPR_MAX_STARVE[15:0]															
	Reset value	0	0	0	0	1	1	1	1									0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0x268	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x26C	DDRCTRL_PERFWR1	W_XACT_RUN_LENGTH[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	W_MAX_STARVE[15:0]															
	Reset value	0	0	0	0	1	1	1	1									0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0x270 - 0x2FC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	DDRCTRL_DBG0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_COLLISION_PAGE_OPT	Res.	Res.	Res.	DIS_WC
	Reset value																												0				0
0x304	DDRCTRL_DBG1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_HIF	DIS_DQ
	Reset value																															0	0
0x308	DDRCTRL_DBGCAM	Res.	Res.	WR_DATA_PIPELINE_EMPTY	RD_DATA_PIPELINE_EMPTY	Res.	DBG_WR_Q_EMPTY	DBG_RD_Q_EMPTY	DBG_STALL	Res.	Res.	Res.	DBG_W_Q_DEPTH[4:0]					Res.	Res.	Res.	DBG_LPR_Q_DEPTH[4:0]					Res.	Res.	Res.	DBG_HPR_Q_DEPTH[4:0]				
	Reset value			0	0		0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x30C	DDRCTRL_DBGCMD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD	ZQ_CALIB_SHOR	Res.	Res.	Res.	RANK0_REFRESH
	Reset value																											0	0				0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x310	DDRCTRL_DBGSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTRLUPD_BUSY	ZQ_CALIB_SHORT_BUSY	Res.	Res.	Res.	RANK0_REFRESH_BUSY
	Reset value																											0	0				0
0x314 - 0x31C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x320	DDRCTRL_SWCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE
	Reset value																																1
0x324	DDRCTRL_SWSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_DONE_ACK
	Reset value																																1
0x328 - 0x368	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x36C	DDRCTRL_POISONCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_CLR	Res.	Res.	Res.	RD_POISON_INTR_EN	Res.	Res.	Res.	RD_POISON_SLVERR_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_CLR	Res.	Res.	Res.	WR_POISON_INTR_EN	Res.	Res.	Res.	WR_POISON_SLVERR_EN
	Reset value								0				1				1								0				1				1
0x370	DDRCTRL_POISONSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_POISON_INTR_1	RD_POISON_INTR_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_POISON_INTR_1	WR_POISON_INTR_0
	Reset value															0	0															0	0
0x374 - 0x3F8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3FC	DDRCTRL_PSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_BUSY_1	WR_PORT_BUSY_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RD_PORT_BUSY_1	RD_PORT_BUSY_0
	Reset value															0	0															0	0
0x400	DDRCTRL_PCCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL_EXP_MODE	Res.	Res.	Res.	PAGEMATCH_LIMIT	Res.	Res.	Res.	GO2CRITICAL_EN
	Reset value																								0				0				0
0x404	DDRCTRL_PCFGR_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDWR_ORDERED_EN	Res.	RD_PORT_PAGEMATCH_EN	RD_PORT_URGENT_EN	RD_PORT_AGING_EN	Res.	Res.	RD_PORT_PRIORITY[9:0]									
	Reset value																0		1	0	0			0	0	0	0	0	0	0	0	0	0
0x408	DDRCTRL_PCFGW_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_PAGEMATCH_EN	WR_PORT_URGENT_EN	WR_PORT_AGING_EN	Res.	Res.	WR_PORT_PRIORITY[9:0]									
	Reset value																		1	0	0			0	0	0	0	0	0	0	0	0	0
0x40C - 0x48C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x490	DDRCTRL_PCTRL_0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_EN
	Reset value																																0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x494	DDRCTRL_PCFGQOS0_0	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_REGION2[1:0]		Res.	Res.	RQOS_MAP_REGION1[1:0]		Res.	Res.	RQOS_MAP_REGION0[1:0]		Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL1[3:0]			
	Reset value							1	0			0	0			0	0					1	1	1	0					0	0	0	0
0x498	DDRCTRL_PCFGQOS1_0	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTR[10:0]											Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTB[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x49C	DDRCTRL_PCFGWQOS0_0	Res.	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_REGION2[1:0]		Res.	Res.	WQOS_MAP_REGION1[1:0]		Res.	Res.	WQOS_MAP_REGION0[1:0]		Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL1[3:0]			
	Reset value							0	0			0	0			0	0					1	1	1	0					0	0	0	0
0x4A0	DDRCTRL_PCFGWQOS1_0	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT2[10:0]											Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT1[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x4A4 - 0x4B0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4B4	DDRCTRL_PCFGR_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDWR_ORDERED_EN	Res.	RD_PORT_PAGEMATCH_EN	RD_PORT_URGENT_EN	RD_PORT_AGING_EN	Res.	Res.	RD_PORT_PRIORITY[9:0]									
	Reset value																0		1	0	0			0	0	0	0	0	0	0	0	0	0
0x4B8	DDRCTRL_PCFGW_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR_PORT_PAGEMATCH_EN	WR_PORT_URGENT_EN	WR_PORT_AGING_EN	Res.	Res.	WR_PORT_PRIORITY[9:0]									
	Reset value																		1	0	0			0	0	0	0	0	0	0	0	0	0
0x4BC - 0x53C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x540	DDRCTRL_PCTRL_1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_EN
	Reset value																																0
0x544	DDRCTRL_PCFGQOS0_1	Res.	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_REGION2[1:0]		Res.	Res.	RQOS_MAP_REGION1[1:0]		Res.	Res.	RQOS_MAP_REGION0[1:0]		Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	RQOS_MAP_LEVEL1[3:0]			
	Reset value							1	0			0	0			0	0					1	1	1	0					0	0	0	0
0x548	DDRCTRL_PCFGQOS1_1	Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTR[10:0]											Res.	Res.	Res.	Res.	Res.	RQOS_MAP_TIMEOUTB[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Table 19. DDRCTRL register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54C	DDRCTRL_PCFGWQOS0_1	Res.	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_REGION2[1:0]		Res.	Res.	WQOS_MAP_REGION1[1:0]		Res.	Res.	WQOS_MAP_REGION0[1:0]		Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL2[3:0]				Res.	Res.	Res.	Res.	WQOS_MAP_LEVEL1[3:0]			
	Reset value							0	0			0	0			0	0					1	1	1	0					0	0	0	0
0x550	DDRCTRL_PCFGWQOS1_1	Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT2[10:0]											Res.	Res.	Res.	Res.	Res.	WQOS_MAP_TIMEOUT1[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

6 TrustZone® address space controller for DDR (TZC)

6.1 Introduction

The TZC filters read/write accesses to the DDR controller according to TrustZone access rights, and according to Non-secure master Address ID (NSAID) on up to 9 programmable regions.

Note: For more information, please refer to Arm TZC-400 TRM.

6.2 TZC features

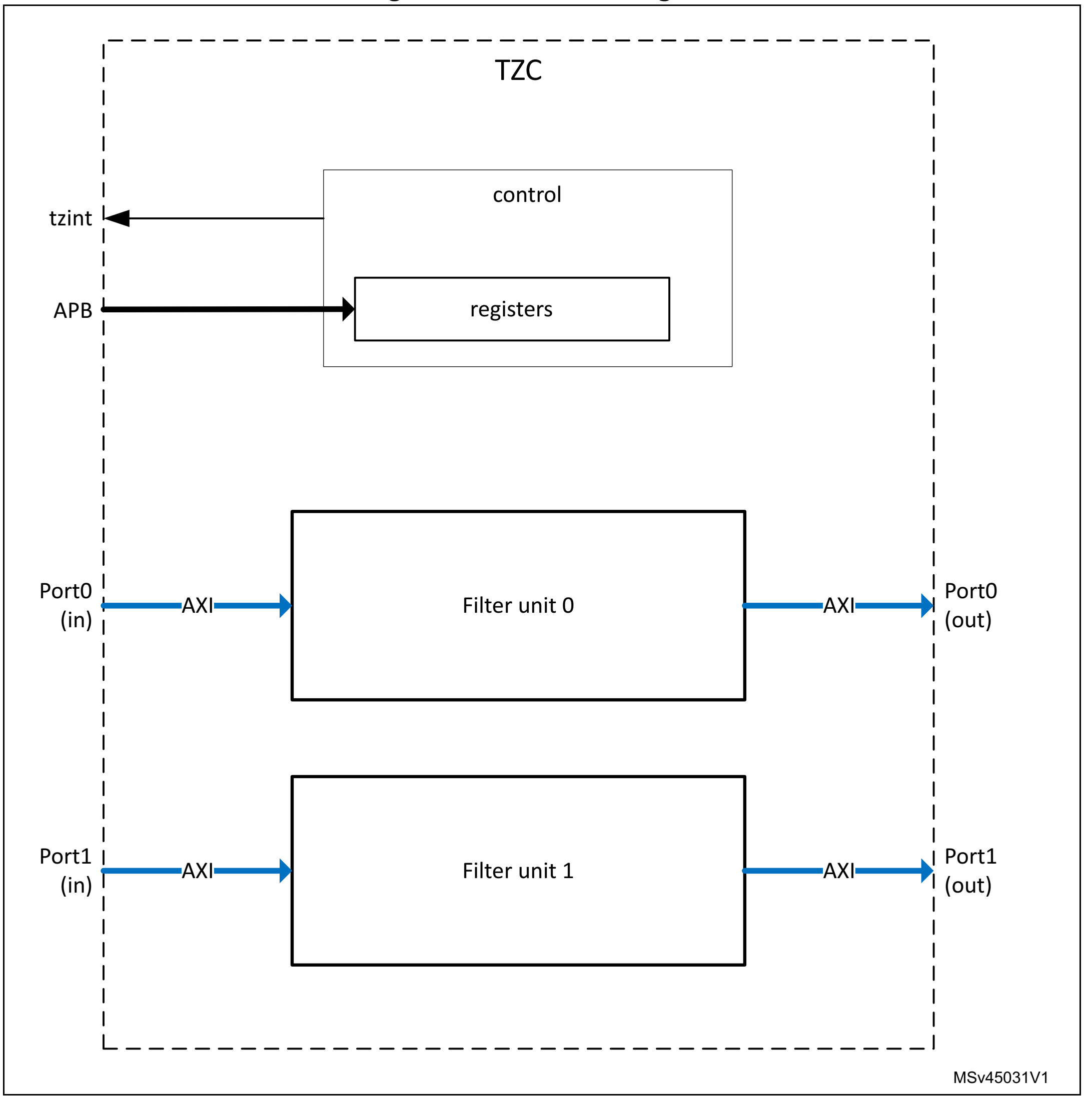
- 32-bit APB4 Interface
- TZC configuration is supported by secure master only
- 2 filter units working concurrently
- 9 regions:
  - Region 0 is always enabled and covers the whole address range
  - Regions 1 to 8 have programmable base/end address and can be assigned one or both filters.
- Secure and non-secure access permissions are programmed per region
- Non-secure accesses are filtered according to NSAID
- Regions controlled by the same filter must not overlap
- Failed permission checks are signaled with an AXI Bus error and/or interrupt
- Dual read access path with:
  - Normal path supporting 256 outstanding transactions
  - Fast path with limited outstanding capabilities using FPID signals
- Gate keeper logic to enable and disable each filter
- Speculative access

Note: Fast path and FPID signals are not used in STM32MP153x. Configured acceptance on FPID is set to minimum (8).

6.3 TZC block diagram

The TZC block diagram is shown in *Figure 8: TZC block diagram*.

Figure 8. TZC block diagram



6.4 TZC functional description

The TZC enforces the access permissions of several regions:

- region 0: always enabled and covers the whole 32-bit address space.
- regions 1 to 8: are fully programmable.

Access permissions are checked against transaction security and the transaction source:

- Transaction security is identified by AXI TrustZone protection signals: awprot[1] and arprot[1].
- The transaction source is identified by the Master NSAID. In the case of a permission fail, the fail ID is reported by the AXI ID on the AXI port. The NSAID and AXI ID are defined according to *Table 26: NSAID and AXI _ID mapping*.

A region must be granted secure access if required, regardless of the settings for non-secure reads and non-secure writes. For example, when permissions are set so that a non-secure read or non-secure write access is permitted, this does not mean that the corresponding secure read or secure write access is also automatically granted.

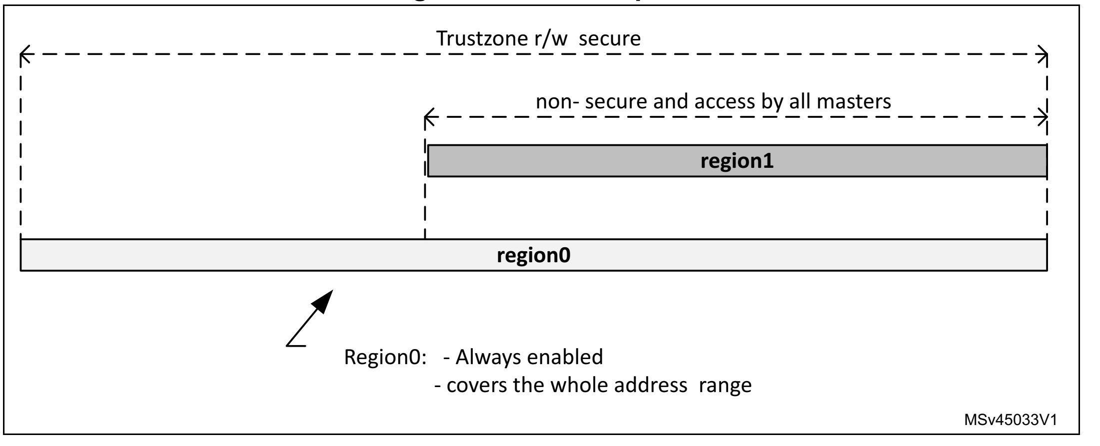
Table 20. NSAID and AXI _ID mapping

Master	NSAID[3:0]	ID_WIDTH	AXI_ID[10:0]
CA7 (CPU0 and 1)	0b0000	AR:6/AW:5	11'b10XXXXXX000
CM4	0b0001	0	11'b00000000001
LTDC	0b0011	4	11'b0100XXXX101
MDMA	0b0101	1	11'b0000000X000
DMA1	0b0110	0	11'b00000000001
DMA2	0b0110	0	11'b00000000001
USBH (EHCI)	0b0111	0	11'b01000000000
USBH (OHCI)	0b0111	0	11'b01000000001
OTG	0b1000	0	11'b00000000001
SDMMC1	0b1001	0	11'b01000000011
SDMMC2	0b1001	0	11'b01000000100
SDMMC3	0b1001	0	11'b00000000001
ETH	0b1010	4	11'b0100XXXX010
DAP	0b1111	0	11'b00000000010

6.5 TZC application programming examples

A simple application example is shown *Figure 9: TZC example 1*:

Figure 9. TZC example 1



In this example the secure world has access to whole DDR range, and all non-secure masters have permission to access address ≥ base1.

It is programmed with:

- region 0
  - always enabled by default
  - covers the full address range by default
  - set region0 secure r/w access permissions
  - set region0 NSAID r/w access permission
  - enable both filters on region0

```
REGION0_ATTRIBUTE.s_wr_en = 1
REGION0_ATTRIBUTE.s_rd_en = 1
REGION0_ID_ACCESS.nsaid_we_en = 0b00000000000
REGION0_ID_ACCESS.nsaid_re_en = 0b00000000000
REGION0_ATTRIBUTE.filter_en = 0b11
```

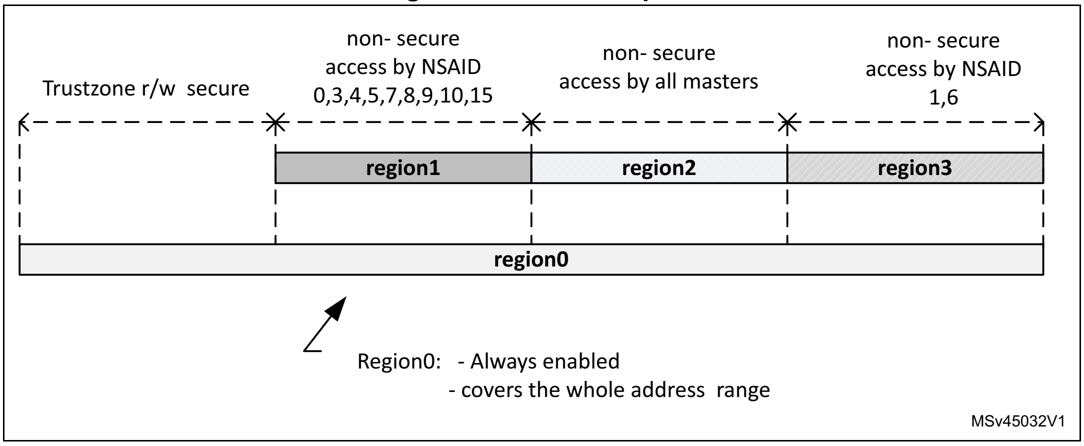
The secure world then defines the other regions (1 up to 8) and opens access to all non-secure NSAIDs.

- region1
  - define region1
  - region1 accessible by secure r/w
  - set region1 NSAID r/w permission for all NSAID
  - enable both filters on region1

```
TZC.REGION_BASE_LOW1.base_address_low = 'base1
REGION1_TOP_LOW.top_address_low = 'base1 + 'size1
REGION1_ATTRIBUTE.s_wr_en = 1
REGION1_ATTRIBUTE.s_rd_en = 1
REGION1_ID_ACCESS.nsaid_we_en = 0xFFFF
REGION1_ID_ACCESS.nsaid_re_en = 0xFFFF
REGION1_ATTRIBUTE.filter_en = 0b11
```

Another application example for the TZC is shown *Figure 10: TZC example 2*:

Figure 10. TZC example 2



In this example the secure world only has access to address ≤ base1 and non-secure masters have selective permissions to region1 and 3; region2 is shared by all non-secure masters.

It is programmed with:

- region 0
  - always enabled by default
  - covers the full address range by default
  - set region0 secure r/w access permissions
  - set region0 NSAID r/w access permission
  - enable both filters on region0

```
REGION0_ATTRIBUTE.s_wr_en = 1
REGION0_ATTRIBUTE.s_rd_en = 1
REGION0_ID_ACCESS.nsaid_we_en = 0x0000
REGION0_ID_ACCESS.nsaid_re_en = 0x0000
REGION0_ATTRIBUTE.filter_en = 0b11
```

The secure world then defines several regions (1 to 3) and opens access to the non-secure NSAID.

- region1
  - define region1
  - region1 non accessible by secure r/w
  - set region1 NSAID r/w permission for: 0,3,4,5,7,8,9,10,15
  - enable both filters on region1
    TZC.REGION_BASE_LOW1.base_address_low = 'base1

```
REGION1_TOP_LOW.top_address_low = 'base1 + 'size1
REGION1_ATTRIBUTE.s_wr_en = 0
REGION1_ATTRIBUTE.s_rd_en = 0
REGION1_ID_ACCESS.nsaid_we_en = 0x87B9
```

```
REGION1_ID_ACCESS.nsaid_re_en = 0x87B9
REGION1_ATTRIBUTE.filter_en = 0b11
```

- region2
  - define region2
  - region2 non accessible by secure r/w
  - set region2 NSAID r/w permission for: 0,1,3,4,5,6,7,8,9,10,15
  - enable both filters on region2

```
REGION2_BASE_LOW.base_address_low = 'base2
REGION2_TOP_LOW.top_address_low = 'base2 + 'size2
REGION2_ATTRIBUTE.s_wr_en = 0
REGION2_ATTRIBUTE.s_rd_en = 0
REGION2_ID_ACCESS.nsaid_we_en = 0x87FB
REGION2_ID_ACCESS.nsaid_re_en = 0x87FB
REGION2_ATTRIBUTE.filter_en = 0b11
```

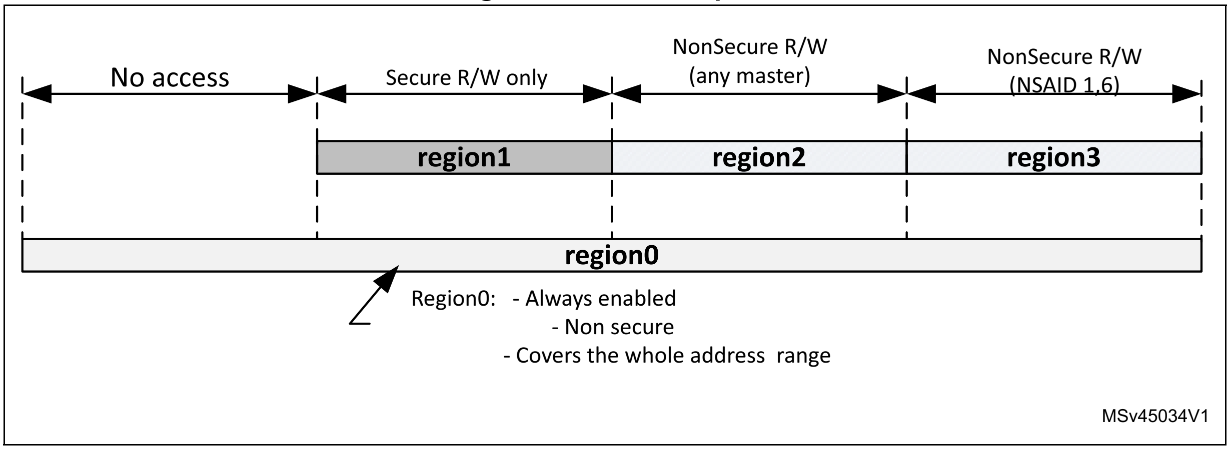
- region3
  - define region3
  - region3 non accessible by secure r/w
  - set region3 NSAID r/w permission for: 1,6
  - enable both filters on region3

```
REGION3_BASE_LOW.base_address_low = 'base3
REGION3_TOP_LOW.top_address_low = 'base3 + 'size3
REGION3_ATTRIBUTE.s_wr_en = 0
REGION3_ATTRIBUTE.s_rd_en = 0
REGION3_ID_ACCESS.nsaid_we_en = 0x0042
REGION3_ID_ACCESS.nsaid_re_en = 0x0042
REGION3_ATTRIBUTE.filter_en = 0b11
```

Note: All shared regions need to be distinct because regions must not overlap.

A third application example for the TZC is shown *Figure 11: TZC example 3*: This example uses region0 as a blanket and secure or non-secure regions are set as holes. This programming model is more convenient for trapping programming errors.

Figure 11. TZC example 3



In this example the secure world only has access to address ≤ base1 and non-secure masters have selective permissions to region2 and 3.

It is programmed with:

- region 0
  - always enabled by default
  - covers the full address range by default
  - set region0 without secure access permissions (default)
  - set region0 without NSAID r/w access permission (default)
  - enable both filters on region0

```
REGION0_ATTRIBUTE.s_wr_en = 0 (default)
REGION0_ATTRIBUTE.s_rd_en = 0 (default)
REGION0_ID_ACCESS.nsaid_we_en = 0x0000 (default)
REGION0_ID_ACCESS.nsaid_re_en = 0x0000 (default)
REGION0_ATTRIBUTE.filter_en = 0b11
```

The secure world then defines several regions (1 to 3) and opens holes for secure and non-secure access

- region1
  - define region1
  - region1 accessible by secure r/w
  - set region1 no NSAID r/w permission
  - enable both filters on region1

```
TZC.REGION_BASE_LOW1.base_address_low = 'base1
REGION1_TOP_LOW.top_address_low = 'base1 + 'size1
REGION1_ATTRIBUTE.s_wr_en = 1
REGION1_ATTRIBUTE.s_rd_en = 1
REGION1_ID_ACCESS.nsaid_we_en = 0x0000
REGION1_ID_ACCESS.nsaid_re_en = 0x0000
REGION1_ATTRIBUTE.filter_en = 0b11
```

- region2
  - define region2
  - region2 non accessible by secure r/w
  - set region2 NSAID r/w permission for all NSAID
  - enable both filters on region2

```
REGION2_BASE_LOW.base_address_low = 'base2
REGION2_TOP_LOW.top_address_low = 'base2 + 'size2
REGION2_ATTRIBUTE.s_wr_en = 0
REGION2_ATTRIBUTE.s_rd_en = 0
REGION2_ID_ACCESS.nsaid_we_en = 0xFFFF
REGION2_ID_ACCESS.nsaid_re_en = 0xFFFF
REGION2_ATTRIBUTE.filter_en = 0b11
```

- region3
  - define region3
  - region3 non accessible by secure r/w
  - set region3 NSAID r/w permission for NSAID 1,6
  - enable both filters on region3

```
REGION3_BASE_LOW.base_address_low = 'base3
```

```
REGION3_TOP_LOW.top_address_low = ‘base3 + ‘size3
REGION3_ATTRIBUTE.s_wr_en = 0
REGION3_ATTRIBUTE.s_rd_en = 0
REGION3_ID_ACCESS.nsaid_we_en = 0x0042
REGION3_ID_ACCESS.nsaid_re_en = 0x0042
REGION3_ATTRIBUTE.filter_en = 0b11
```

6.5.1 DMA requests

*Note: TZC does **not** support DMA interface.*

6.5.2 Interrupts

*Note: TZC is asserting interrupt: **tzcint** when some security checks failed.*

6.6 TZC registers

TZC registers are accessed only by words (32-bit).

6.6.1 TZC configuration register (TZC_BUILD_CONFIG)

Address offset: 0x000

Reset value: 0x0100 1F08

Provides information about TZC configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	NO_OF_FILTERS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADDRESS_WIDTH[5:0]						Res.	Res.	Res.	NO_OF_REGIONS[4:0]				
		r	r	r	r	r	r				r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **NO_OF_FILTERS[1:0]**: Number of filters
0x1: 2 filters
Others: Reserved

Bits 23:14 Reserved, must be kept at reset value.

Bits 13:8 **ADDRESS_WIDTH[5:0]**: Address width
0x1F: 32-bit address width
Others: Reserved

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NO_OF_REGIONS[4:0]**: Number fo regions
0x8: 9 regions
Others: Reserved

6.6.2 TZC action register (TZC_ACTION)

Address offset: 0x004

Reset value: 0x0000 0000

Controls interrupt and bus error response behavior when regions permission failures occur.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACTION_VALUE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **REACTION_VALUE[1:0]**: Permission failure reaction
Controls how TZC signals region permission failure.
0: set tzcint low and issue OKAY on the bus
1: set tzcint low and issue DECERR on the bus
2: set tzcint high and issue OKAY on the bus
3: set tzcint high and issue DECERR on the bus

6.6.3 TZC gate keeper register (TZC_GATE_KEEPER)

Address offset: 0x008

Reset value: 0x0000 0000

Provides control and status for the gate keeper in the implemented filter units.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPENSTAT[1:0]	
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPENREQ[1:0]	
														rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **OPENSTAT[1:0]**: Gate keeper status for each filter
Bit 17 is for filter1, bit 16 is for filter0
0: filter is opened
1: filter is closed

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **OPENREQ[1:0]**: Gate keeper open request
Bit 1 is for filter1, bit 0 is for filter0.
0: request filter to open
1: request filter to close

6.6.4 TZC speculation control register (TZC_SPECULATION_CTRL)

Address offset: 0x00C

Reset value: 0x0000 0000

Controls read and write access speculation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITE SPEC_ DISA BLE	READ SPEC_ DISA BLE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **WRITESPEC_DISABLE**: Write access speculation disable
0: write access speculation enabled
1: write access speculation disabled

Bit 0 **READSPEC_DISABLE**: Read access speculation disable
0: read access speculation enabled
1: read access speculation disabled

6.6.5 TZC interrupt status register (TZC_INT_STATUS)

Address offset: 0x010

Reset value: 0x0000 0000

Contains the status of the interrupt signal, TZCINT, that reports access security violations or region overlap errors.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVERLAP[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OVERRUN[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	STATUS[1:0]	
						r	r							r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **OVERLAP[1:0]**: Overlap violation for each filter
Bit 17 is for filter 1, bit 16 is for filter 0.
0: interrupt is not asserted
1: interrupt asserted and waiting to be cleared

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **OVERRUN[1:0]**: Permission failure overrun
Two or more regions permission failures for each filter. Bit 9 is for filter 1, bit 8 is for filter 0.
0: interrupt is not asserted
1: interrupt asserted and waiting to be cleared

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **STATUS[1:0]**: Interrupt status for each filter
Bit 1 is for filter 1, bit 0 is for filter0.
0: interrupt is not asserted
1: interrupt asserted and waiting to be cleared

6.6.6 TZC interrupt clear register (TZC_INT_CLEAR)

Address offset: 0x014

Reset value: 0x0000 0000

Interrupt clear for each filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLEAR[1:0]	
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **CLEAR[1:0]**: Filter interrupt clear
Write “1” to clear interrupt for each filter.
bit 1: for filter 1
bit 0: for filter 0

6.6.7 TZC fail address low register x (TZC_FAIL_ADDRESS_LOWx)

Address offset: 0x020 + 0x010 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Address low bits of the first failed access in the associated filter (0 to 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR_STATUS_LOW[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR_STATUS_LOW[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ADDR_STATUS_LOW[31:0]**: Fail address low bits
Low 32 address bits of the first failed access permission check in the associated filter (0 to 1).

6.6.8 TZC fail address high register x (TZC_FAIL_ADDRESS_HIGHx)

Address offset: 0x024 + 0x010 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Address high bit of the first failed access in the associated filter (0 to 1).

Note: Not used with 32bit address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

6.6.9 TZC fail control register x (TZC_FAIL_CONTROLx)

Address offset: 0x028 + 0x010 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Status information about the first access that failed a region permission check in the associated filter (0 to 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRECTION	Res.	Res.	NON_SECURE	PRIVILEGE	Res.	Res.	Res.	Res.
							r			r	r				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIRECTION**: Access failure direction
- 0: Read access failure
- 1: Write access failure

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **NON_SECURE**: Non-secure access failure
- 0: Secure access failure
- 1: Non-secure access failure

Bit 20 **PRIVILEGE**: Privilege access failure
- 0: Unprivileged access failure
- 1: Privileged access failure

Bits 19:0 Reserved, must be kept at reset value.

6.6.10 TZC fail ID register x (TZC_FAIL_IDx)

Address offset: 0x02C + 0x010 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Contains the master AXI ARID or AWID of the first access that failed a region permission check in the associated filter unit. This occurs even if the ACTION register is set to not drive the interrupt signal.

Note: AXI ID mapping is described in Table 26: NSAID and AXI _ID mapping.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	ID[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **ID[10:0]**: AXI fail ID
Return the AXI ID of the first fail.

6.6.11 TZC region 0 base address low register (TZC_REGION_BASE_LOW0)

Address offset: 0x100

Reset value: 0x0000 0000

Base address low for region 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASE_ADDRESS_LOW[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BASE_ADDRESS_LOW[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

Bits 31:12 **BASE_ADDRESS_LOW[19:0]**: base address bits[31:12] for region 0
0x00000: fixed value for region 0
Others: Reserved

Bits 11:0 Reserved, must be kept at reset value.

6.6.12 TZC region x base address high register (TZC_REGION_BASE_HIGHx)

Address offset: 0x104 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

Base address high are not used with 32-bit address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

6.6.13 TZC region 0 top address low register (TZC_REGION_TOP_LOW0)

Address offset: 0x108

Reset value: 0xFFFF FFFF

Top address bits [31:12] for region 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP_ADDRESS_LOW[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOP_ADDRESS_LOW[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

Bits 31:12 **TOP_ADDRESS_LOW[19:0]**: Top address bits [31:12] of region 0
0xFFFFF: fixed value for region 0
Others: Reserved

Bits 11:0 Reserved, must be kept at reset value.

6.6.14 TZC region x top address high register (TZC_REGION_TOP_HIGHx)

Address offset: 0x10C + 0x20 * x, (x= 0 to 8)

Reset value: 0x0000 0000

Top address high of region are not used with 32-bit address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

6.6.15 TZC region 0 attribute register (TZC_REGION_ATTRIBUTE0)

Address offset: 0x110

Reset value: 0x0000 0003

Region 0 attributes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
														r	r

Bit 31 **S_WR_EN**: Secure global write enable
0: secure write to the region are not allowed
1: permit secure write into the region

Bit 30 **S_RD_EN**: Secure global read enable
0: secure read to the region are not allowed
1: permit secure read into the region

Bits 29:2 Reserved, must be kept at reset value.

Bits 1:0 **FILTER_EN[1:0]**: Region enable for each filter
Bit 1 is for filter 1
Bit 0 is for filter 0
0: Reserved
1: enable filter for the region

Note: For region 0 the FILTER_EN bits are '1" and cannot be modified.

6.6.16 TZC region x ID access register (TZC_REGION_ID_ACCESSx)

Address offset: 0x114 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

Region non-secure access based on NSAID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSAID_WR_EN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSAID_RD_EN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **NSAID_WR_EN[15:0]**: Region write enable for each NSAID
Bit 16 is associated to NSAID = 0 .. bit 31 is associated to NSAID = 15
0: forbids write non-secure to the region for this NSAID
1: permits write non-secure to the region for this NSAID

Bits 15:0 **NSAID_RD_EN[15:0]**: Region read enable for each NSAID
Bit 0 is associated to NSAID = 0 ... bit 15 is associated to NSAID = 15.
0: forbids read non-secure to the region for this NSAID
1: permits read non-secure to the region for this NSAID

6.6.17 TZC region x base address low register (TZC_REGION_BASE_LOWx)

Address offset: 0x120 + 0x20 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

Base address low for regions 1 to 8.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASE_ADDRESS_LOW[19:4]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE_ADDRESS_LOW[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:12 **BASE_ADDRESS_LOW[19:0]**: Base address bits[31:12] for region x

Bits 11:0 Reserved, must be kept at reset value.

6.6.18 TZC region x top address low register (TZC_REGION_TOP_LOWx)

Address offset: 0x128 + 0x20 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0FFF

Top address bits [31:12] for region x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP_ADDRESS_LOW[19:4]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOP_ADDRESS_LOW[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:12 **TOP_ADDRESS_LOW[19:0]**: Top address bits [31:12] of region x

Bits 11:0 Reserved, must be kept at reset value.

6.6.19 TZC region x attribute register (TZC_REGION_ATTRIBUTEx)

Address offset: 0x130 + 0x20 * (x-1), (x = 1 to 8)

Reset value: 0x0000 0000

Region x attributes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
														rw	rw

Bit 31 **S_WR_EN**: Secure global write enable
0: secure write to the region are not allowed
1: permit secure write into the region

Bit 30 **S_RD_EN**: Secure global read enable
0: secure read to the region are not allowed
1: permit secure read into the region

Bits 29:2 Reserved, must be kept at reset value.

Bits 1:0 **FILTER_EN**[1:0]: Region enable for each filter
Bit 1 is for filter 1
Bit 0: is for filter 0
0: disable filter for the region
1: enable filter for the region

6.6.20 TZC peripheral ID 4 register (TZC_PID4)

Address offset: 0xFD0

Reset value: 0x0000 0004

Peripheral ID 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_4[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_4[7:0]**: Peripheral ID 4

6.6.21 TZC peripheral ID 5 register (TZC_PID5)

Address offset: 0xFD4

Reset value: 0x0000 0000

Peripheral ID 5.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_5[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_5[7:0]**: Peripheral ID 5

6.6.22 TZC peripheral ID 6 register (TZC_PID6)

Address offset: 0xFD8

Reset value: 0x0000 0000

Peripheral ID 6.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_6[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_6[7:0]**: Peripheral ID 6

6.6.23 TZC peripheral ID 7 register (TZC_PID7)

Address offset: 0xFDC

Reset value: 0x0000 0000

Peripheral ID 7.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_7[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_7[7:0]**: Peripheral ID 7

6.6.24 TZC peripheral ID 0 register (TZC_PID0)

Address offset: 0xFE0

Reset value: 0x0000 0060

Peripheral ID 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_0[7:0]**: Peripheral ID 0

6.6.25 TZC peripheral ID 1 register (TZC_PID1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

Peripheral ID 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_1[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_1[7:0]**: Peripheral ID 1

6.6.26 TZC peripheral ID 2 register (TZC_PID2)

Address offset: 0xFE8

Reset value: 0x0000 002B

Peripheral ID 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_2[7:0]**: Peripheral ID 2

6.6.27 TZC peripheral ID 3 register (TZC_PID3)

Address offset: 0xFEC

Reset value: 0x0000 0000

Peripheral ID 3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PER_ID_3[7:0]**: Peripheral ID 3

6.6.28 TZC component ID 0 register (TZC_CID0)

Address offset: 0xFF0

Reset value: 0x0000 000D

Component ID 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COMP_ID_0[7:0]**: Component ID 0

6.6.29 TZC component ID 1 register (TZC_CID1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

Component ID 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_1[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COMP_ID_1[7:0]**: Component ID 0

6.6.30 TZC component ID 2 register (TZC_CID2)

Address offset: 0xFF8

Reset value: 0x0000 0005

Component ID 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COMP_ID_2[7:0]**: Component ID 2

6.6.31 TZC component ID 3 register (TZC_CID3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

Component ID 3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COMP_ID_3[7:0]**: Component ID 3

6.6.32 TZC register map

Table 21. TZC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TZC_BUILD_ CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	NO_OF_FILTERS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRESS_WIDTH[5:0]						Res.	Res.	Res.	NO_OF_REGIONS[4:0]				
	Reset value							0	1											0	1	1	1	1	1				0	1	0	0	0
0x004	TZC_ACTION	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACTION_VALUE[1:0]	
	Reset value																															0	0
0x008	TZC_GATE_ KEEPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPENSTAT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPENREQ[1:0]	
	Reset value															0	0															0	0
0x00C	TZC_ SPECULATION_ CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRITESPEC_DISABLE	READSPEC_DISABLE
	Reset value																															0	0
0x010	TZC_INT_STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVERLAP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	OVERRUN[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	STATUS[1:0]	
	Reset value															0	0							0	0							0	0
0x014	TZC_INT_CLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLEAR[1:0]	
	Reset value																															0	0
0x018 - 0x01C	Reserved	Res.																															
0x020	TZC_FAIL_ ADDRESS_LOW0	ADDR_STATUS_LOW[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TZC_FAIL_ ADDRESS_HIGH0	FAIL_ADDRESS_HIGH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21. TZC register map and reset values (continued)

Offset	Register name reset value	Register size																															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	**TZC_FAIL_CONTROL0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRECTION	Res.	Res.	NON_SECURE	PRIVILEGE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0			0	0																				
0x02C	**TZC_FAIL_ID0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x030 - 0x0FC	Reserved	Res.																															
0x100	**TZC_REGION_BASE_LOW0**	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x104	**TZC_REGION_BASE_HIGH0**	REGION_BASE_HIGH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	**TZC_REGION_TOP_LOW0**	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1												
0x10C	**TZC_REGION_TOP_HIGH0**	REGION_TOP_HIGH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110	**TZC_TREGION_ATTRIBUTE0**	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0	0																												1	1
0x114	**TZC_REGION_ID_ACCESS0**	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118 - 0x11C	Reserved	Res.																															
0x120	**TZC_REGION_BASE_LOW1**	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x124	**TZC_REGION_BASE_HIGH1**	REGION_BASE_HIGH1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x128	**TZC_REGION_TOP_LOW1**	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x12C	**TZC_REGION_TOP_HIGH1**	REGION_TOP_HIGH1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x130	**TZC_REGION_ATTRIBUTE1**	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0

Table 21. TZC register map and reset values (continued)

Offset	Register name reset value	Register size																															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x134	TZC_REGION_ID_ACCESS1	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x138 - 0x13C	Reserved	Res.																															
0x140	TZC_REGION_BASE_LOW2	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x144	TZC_REGION_BASE_HIGH2	REGION_BASE_HIGH2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148	TZC_REGION_TOP_LOW2	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x14C	TZC_REGION_TOP_HIGH2	REGION_TOP_HIGH2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	TZC_REGION_ATTRIBUTE2	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0
0x154	TZC_REGION_ID_ACCESS2	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158 - 0x15C	Reserved	Res.																															
0x160	TZC_REGION_BASE_LOW3	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x164	TZC_REGION_BASE_HIGH3	REGION_BASE_HIGH3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x168	TZC_REGION_TOP_LOW3	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x16C	TZC_REGION_TOP_HIGH3	REGION_TOP_HIGH3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x170	TZC_REGION_ATTRIBUTE3	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0
0x174	TZC_REGION_ID_ACCESS3	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x178 - 0x17C	Reserved	Res.																															
0x180	TZC_REGION_BASE_LOW4	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												

Table 21. TZC register map and reset values (continued)

Offset	Register name reset value	Register size																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184	TZC_REGION_ BASE_HIGH4	REGION_BASE_HIGH4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	TZC_REGION_ TOP_LOW4	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x18C	TZC_REGION_ TOP_HIGH4	REGION_TOP_HIGH4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	TZC_REGION_ ATTRIBUTE4	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0
0x194	TZC_REGION_ID_ ACCESS4	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198 - 0x19C	Reserved	Res.																															
0x1A0	TZC_REGION_ BASE_LOW5	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1A4	TZC_REGION_ BASE_HIGH5	REGION_BASE_HIGH5[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A8	TZC_REGION_ TOP_LOW5	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1AC	TZC_REGION_ TOP_HIGH5	REGION_TOP_HIGH5[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1B0	TZC_REGION_ ATTRIBUTE5	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0
0x1B4	TZC_REGION_ID_ ACCESS5	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1B8 - 0x1BC	Reserved	Res.																															
0x1C0	TZC_REGION_ BASE_LOW6	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1C4	TZC_REGION_ BASE_HIGH6	REGION_BASE_HIGH6[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C8	TZC_REGION_ TOP_LOW6	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1CC	TZC_REGION_ TOP_HIGH6	REGION_TOP_HIGH6[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 21. TZC register map and reset values (continued)

Offset	Register name reset value	Register size																															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1D0	TZC_REGION_ATTRIBUTE6	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0
0x1D4	TZC_REGION_ID_ACCESS6	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D8 - 0x1DC	Reserved	Res.																															
0x1E0	TZC_REGION_BASE_LOW7	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1E4	TZC_REGION_BASE_HIGH7	REGION_BASE_HIGH7[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1E8	TZC_REGION_TOP_LOW7	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1EC	TZC_REGION_TOP_HIGH7	REGION_TOP_HIGH7[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1F0	TZC_REGION_ATTRIBUTE7	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0
0x1F4	TZC_REGION_ID_ACCESS7	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1F8 - 0x1FC	Reserved	Res.																															
0x200	TZC_REGION_BASE_LOW8	BASE_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x204	TZC_REGION_BASE_HIGH8	REGION_BASE_HIGH8[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	TZC_REGION_TOP_LOW8	TOP_ADDRESS_LOW[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x20C	TZC_REGION_TOP_HIGH8	REGION_TOP_HIGH8[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	TZC_REGION_ATTRIBUTE8	S_WR_EN	S_RD_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FILTER_EN[1:0]	
	Reset value	0	0																													0	0

Table 21. TZC register map and reset values (continued)

Offset	Register name reset value	Register size																															
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x214	TZC_REGION_ID_ACCESS8	NSAID_WR_EN[15:0]																NSAID_RD_EN[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218 - 0xFCC	Reserved	Res.																															
0xFD0	TZC_PID4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_4[7:0]							
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	TZC_PID5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_5[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFD8	TZC_DRASC_PID6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_6[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFDC	TZC_PID7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_7[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE0	TZC_PID0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_0[7:0]							
	Reset value																									0	1	1	0	0	0	0	0
0xFE4	TZC_PID1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_1[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	TZC_DRASC_PID2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_2[7:0]							
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	TZC_PID3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PER_ID_3[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TZC_CID0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TZC_CID1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_1[7:0]							
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	TZC_CID2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_2[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TZC_CID3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP_ID_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3 on page 101* for the register boundary addresses.

7 DDR physical interface control (DDRPHYC)

7.1 DDRPHYC features

- 32-bit DDR PHY compatible with JEDEC Standards for DDR3/3L, LPDDR2 and LPDDR3 SDRAMs.
- Dual width support: 16-bit (half-width with 2-byte lanes) or 32-bit (full-width with 4-byte lanes) DDRPHYC permits operating with SDRAMs narrower than the implemented data width.
- Operating up to 533 MHz (1066 MT/s) with DDR3/3L.
- Operating up to 533 MHz (1066 MT/s) with LPDDR2/3.
- PHY Utility Block (PUBL) to support the PHY initialization, control and calibrations.
- DFI 2.1 compliant interface to the DDRCTRL controller.
- Programmable output and ODT impedance with dynamic PVT compensation.
- Embedded dynamic drift detection in the PHY to facilitate dynamic drift compensation by the controller.
- Utilizes master and slave DLLs for a precise timing management.
- Lane-based architecture with 4-byte lanes + 1 command/address lane.
- Test modes supporting the IDDq and DLL characterization.

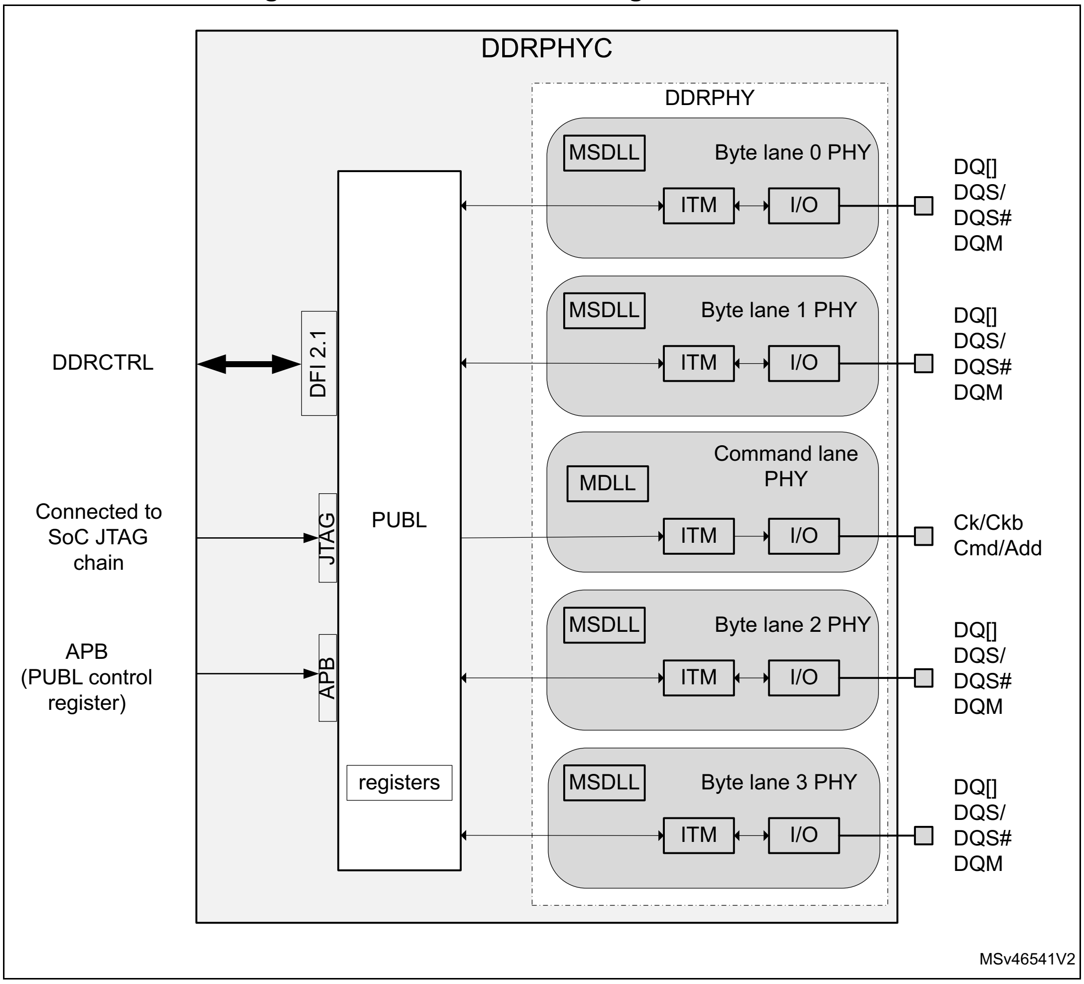
7.2 DDRPHYC block diagram

The DDRPHYC block diagram is shown in *Figure 12*.

The DDRPHYC is composed of:

- A DDRPHY physical interface which is structured into one command lane and 4-byte lanes and used to generate signals with accurate timings at the SDRAM interface.
- A PUBL (PHY Utility Block Logic) which is used to service the PHY and controlled via the APB and JTAG interfaces.

Figure 12. DDRPHYC block diagram



7.3 DDRPHYC SDRAM interface

The DDRPHYC SDRAM interfaces are listed in *Table 22*.

Table 22. DDRPHYC I/O list

Pin name	DDR3/3L	LPDDR2/3	Description
DDR_DQ31 to DDR_DQ24	Yes	Yes	Byte lane 3 DQ
DDR_DQS3P	Yes	Yes	Byte lane 3 DQS
DDR_DQS3N	Yes	Yes	Byte lane 3 DQS#
DDR_DQM3	Yes	Yes	Byte lane 3 DQM
DDR_DQ23 to DDR_DQ16	Yes	Yes	Byte lane 2 DQ
DDR_DQS2P	Yes	Yes	Byte lane 2 DQS
DDR_DQS2N	Yes	Yes	Byte lane 2 DQS#
DDR_DQM2	Yes	Yes	Byte lane 2 DQM

Table 22. DDRPHYC I/O list (continued)

Pin name	DDR3/3L	LPDDR2/3	Description
DDR_DQ15 to DDR_DQ8	Yes	Yes	Byte lane 1 DQ
DDR_DQS1P	Yes	Yes	Byte lane 1 DQS
DDR_DQS1N	Yes	Yes	Byte lane 1 DQS#
DDR_DQM1	Yes	Yes	Byte lane 1 DQM
DDR_DQ7 to DDR_DQ0	Yes	Yes	Byte lane 0 DQ
DDR_DQS0P	Yes	Yes	Byte lane 0 DQS
DDR_DQS0N	Yes	Yes	Byte lane 0 DQS#
DDR_DQM0	Yes	Yes	Byte lane 0 DQM
DDR_A15 to DDR_A0	Yes	Yes(1)	Row/column address / CA[] pins for LPDDR2/3
DDR_BA2 to DDR_BA0	Yes	no	Bank address
DDR_CKE	Yes	Yes	CKE
DDR_CLKP	Yes	Yes	CK
DDR_CLKN	Yes	Yes	CK#
DDR_RASN	Yes	no	RAS#
DDR_CASN	Yes	no	CAS#
DDR_WEN	Yes	no	WE#
DDR_CSN	Yes	Yes	CS#
DDR_ODT	Yes	no	ODT
DDR_RESETN	Yes	no	RESET#
DDR_VREF1 to DDR_VREF3	Yes	Yes(2)	Connected to Vref= VDDQ/2
DDR_ZQ	Yes	Yes	Connected to external R =240 Ω ±1%
DDR_DTO0	Test	Test	Reserved for DLL test purposes
DDR_DTO1	Test	Test	Reserved for DLL test purposes
DDR_ATO	Test	Test	Reserved for DLL test purposes

1. For LPDDR2/3: DDR_A[9:0] are used as CA[9:0] and DDR_A[15:10] are not used.
2. DDR_VREF1: command lane reference, DDR_VREF2: byte lane 3 and 1 common reference, DDR_VREF3: byte lane 2 and 0 common reference.

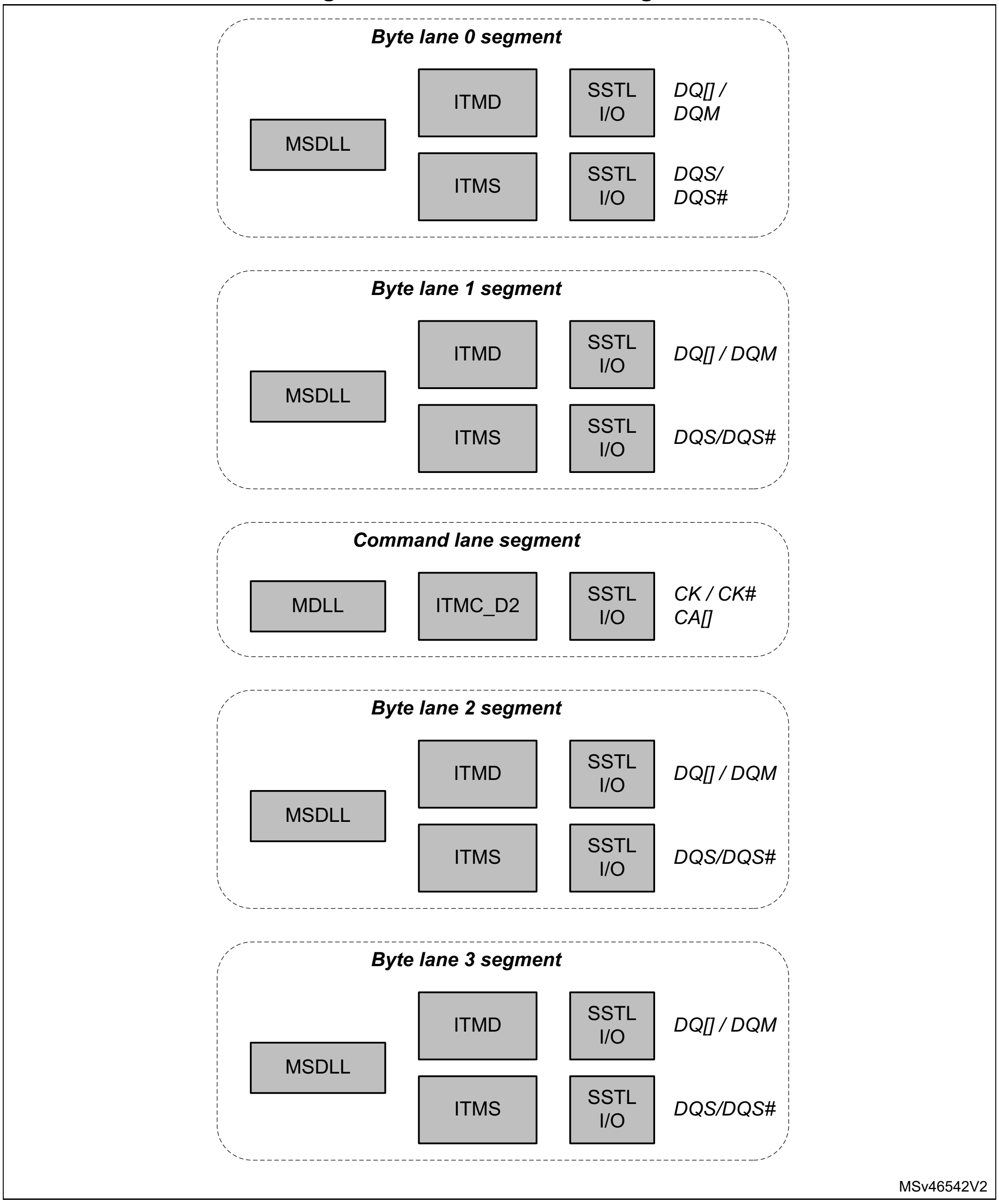
7.4 DDRPHY (DDR multiPHY IP core)

7.4.1 DDRPHY overview

The DDRPHY diagram is shown in *Figure 13*. It is implemented with a structured layout into 4-byte lanes and one address/command lane with:

- MDLL and MSDLLs to provide accurate timings with 90 deg phase shifts
- ITM: Interface timing modules, with the following types:
  - ITMD for DQ/DM signals,
  - ITMS for DQS/DQS# signals,
  - ITMC_D2 for address/command and CK/CK# signals.
- SSTL I/O’s.

Figure 13. DDRPHY block diagram



7.4.2 Delay locked loop (DLL)

The DLLs are shown in *Figure 14*

The Master DLL (MDLL) uses the input (clk_in) to generate four clock outputs, each delayed in quarter clock cycle (90°) increments. These four clock phases (clk_0, clk_90, clk_180, clk_270) are generated with very high accuracy and low jitter across a wide range of frequencies.

The Mast-Slave DLL (MSDLL) is composed of a MDLL and a slave delay pair.

The slave delay pair uses a timing reference from MDLL to provide a highly accurate 90º delay to the DQS/DQS# inputs and generate dqs_90 and dqsb_90 respectively.

DLL key features:

- Master/Slave DLL architecture for complex timing while reducing total power consumption with:
  - Master DLL for SDRAM command / address interface and general host timing.
  - Master-Slave DLL for SDRAM data lanes (DQ/DQS) for write and read data capture.
- Operating range from 100 MHz to 533 MHz.
- Optional: DLL bypass mode for LPDDR suitable for 10 MHz to 200 MHz and for low speed functional and IDDQ tests.
- Multi-phase outputs: 0deg, 90deg, 180deg 270deg.
- Extensive DLL control for fine-tuning system timings.
- Analog and digital DLL test outputs for device characterization.

Figure 14. MDLL block diagram

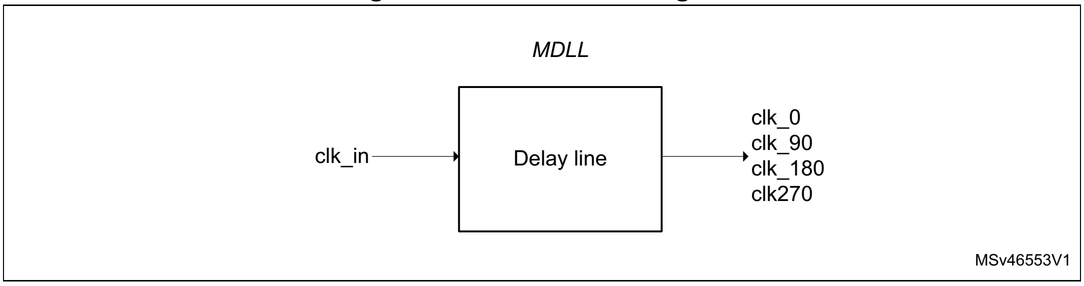
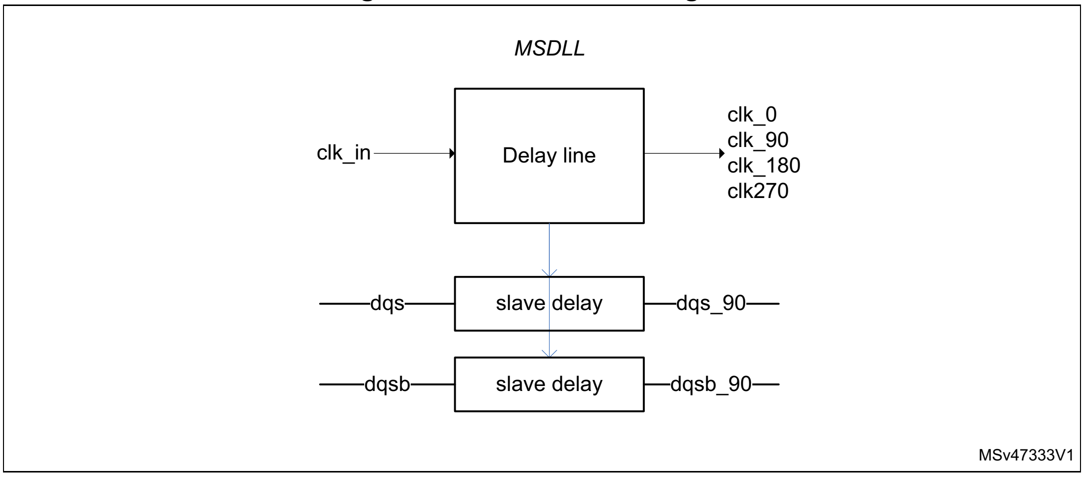


Figure 15. MSDLL block diagram



The DLL settings are controlled by a control bus dllctrl[51:0] described in *Table 23: MSDLL control bits*.

Table 23. MSDLL control bits

Control bus	Signal name	MSDLL PUBL register	MDLL PUBL register	Default	Description
dllctrl[51]	lltm	DXnDLLCR[19]	N/A	0b0	Locked loopback test mode select: 0: normal, 1: Loopback
dllctrl[50]	test_hizb_a	DXnDLLCR[18]	ACDLLCR[18]	0b0	Analog test output tri-state control 0: hiZ, 1: analog low Z
dllctrl[49:46]	sl_phase_trm[3:0]	DXnDLLCR[17:14]	N/A	0b0000	Slave phase lock trim (4 bits encoded 18 deg step from 36 to 144deg)
dllctrl[45:44]	sl_bypass_start_up[1:0]	DXnDLLCR[13:12]	N/A	0b00	Slave auto-startup bypass
dllctrl[43:38]	fb_trm[5:0]	DXnDLLCR[11:6]	ACDLLCR[11:6]	0b000000	Master feedback delay adjust. fb_trm[5:3]: feed forward delay, fb_trm[2:0]: feedback delay
dllctrl[37:32]	sl_fb_trm[5:0]	DXnDLLCR[5:0]	N/A	0b000000	Slave feedback delay adjust (fb_trm[5:3]: ffd, fb_trm[2:0]: fbk)
dllctrl[29]	lock_det_en	DLLGCR[29]	N/A	0b0	Lock detector enable
dllctrl[28:27]	fdtrm_sl[1:0]	DLLGCR[28:27]	DLLGCR[28:27]	0b00	Slave bypass fixed delay trim (-10%, 0, +10%, +20%)
dllctrl[26:24]	sl_bias_trm[6:4]	DLLGCR[26:24]	DLLGCR[26:24]	0b011	Slave bias generator control voltage trim: VC
dllctrl[23]	bps200	DLLGCR[23]	DLLGCR[23]	0b0	Bypass frequency select (0: 0-100MHz, 1:0-200MHz)
dllctrl[22:20]	sl_bias_trm[2:0]	DLLGCR[22:20]	DLLGCR[22:20]	0b111	Slave bias generator frequency trim: Fmax
dllctrl[18:16]	bias_trm[6:4]	DLLGCR[18:16]	DLLGCR[18:16]	0b011	Master bias generator control voltage trim: VC
{dllctrl[19],dllctrl[15]}	fdtrm[1:0]	DLLGCR[19],DLLGCR[15]	DLLGCR[19],DLLGCR[15]	0b00	Master bypass fixed delay trim (-10%, 0, +10%, +20%)
dllctrl[14:12]	bias_trm[2:0]	DLLGCR[14:12]	DLLGCR[14:12]	0b111	Master bias generator frequency trim: Fmax
dllctrl[11]	test_ctrl_switch	DLLGCR[11]	N/A	0b0	Test control: Switch for analog and digital. 0: master, 1: slave
dllctrl[10:9]	test_ctrl_a[1:0]	DLLGCR[10:9]	DLLGCR[10:9]	0b00	Test control: selects the analog signal at ana test
dllctrl[8:6]	test_ctrl_d[2:0]	DLLGCR[8:6]	DLLGCR[8:6]	0b000	Test control: selects the digital signal at dig test
dllctrl[5]	test_ctrl_en	DLLGCR[5]	DLLGCR[5]	0b0	Test control enable for analog and digital test
dllctrl[4:2]	ipump_trm[2:0]	DLLGCR[4:2]	DLLGCR[4:2]	0b000	Charge pump current trim (0: max current)

7.4.3 Interface timing modules (ITMs)

The ITMs are used to generate signals with precise timing:

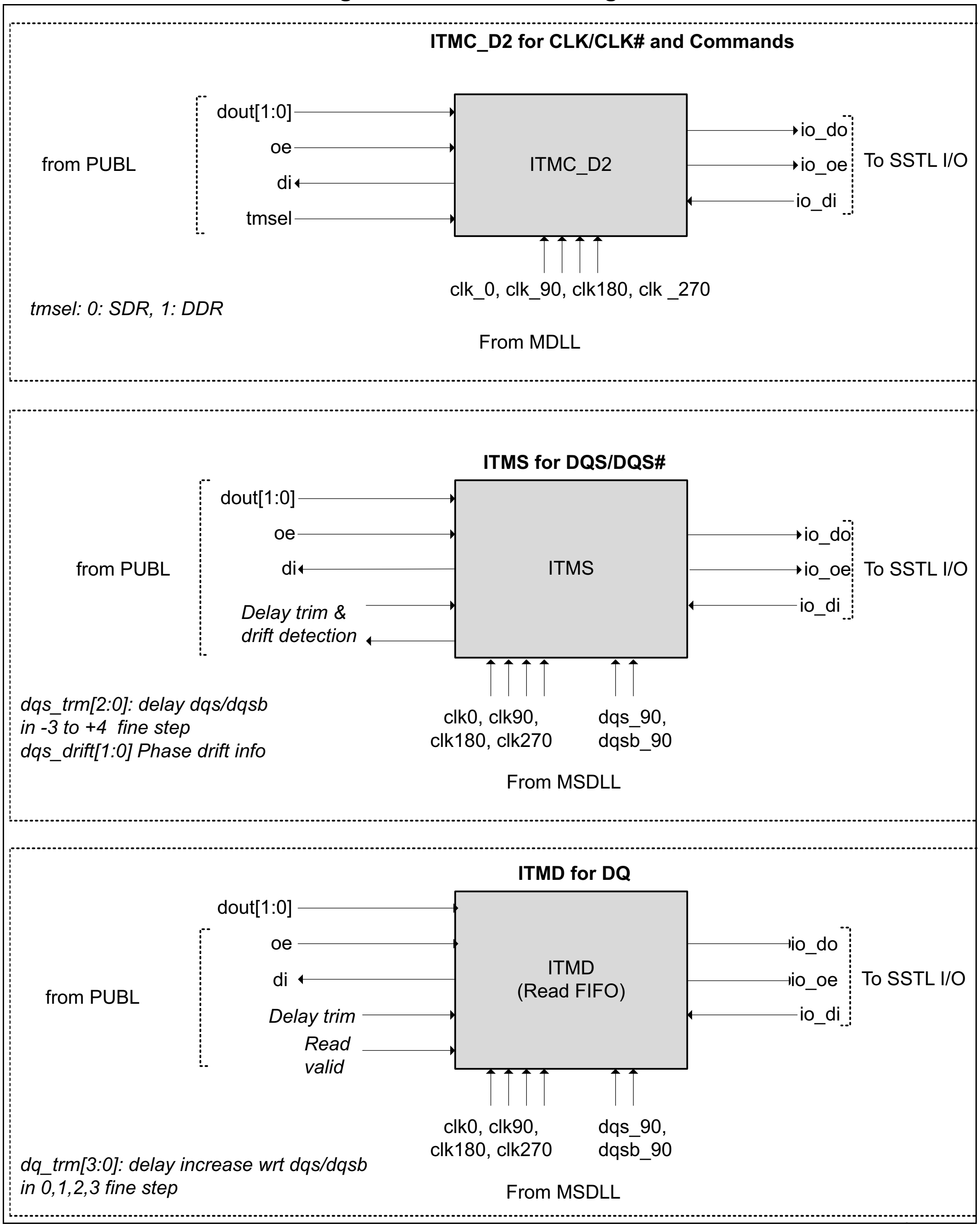
- 90deg phase delay of CK/CK# for the command / address bus
- 90deg phase delay of DQS/DQS# for write DQ/DQM byte lane outputs
- Sampling of read DQ inputs on a 90 deg phase delay of incoming DQS/DQS#

There are three types of ITM:

- ITMC_D2 for clock, address and command lanes.
- ITMS for DQS/ DQS# differential strobes.
- ITMD for DQ, DQM bytre lanes.

The ITMs are shown in *Figure 16*.

Figure 16. ITMs block diagram



The ITMC-D2 does the SDR to DDR signaling conversion for LPDDR2/3 (tmsel =1).

The ITMC_D2 connections to set timings according to SDRAM type are shown in *Figure 17* and *Figure 18*.

Figure 17. ITMC_D2 for LPDDR2/3

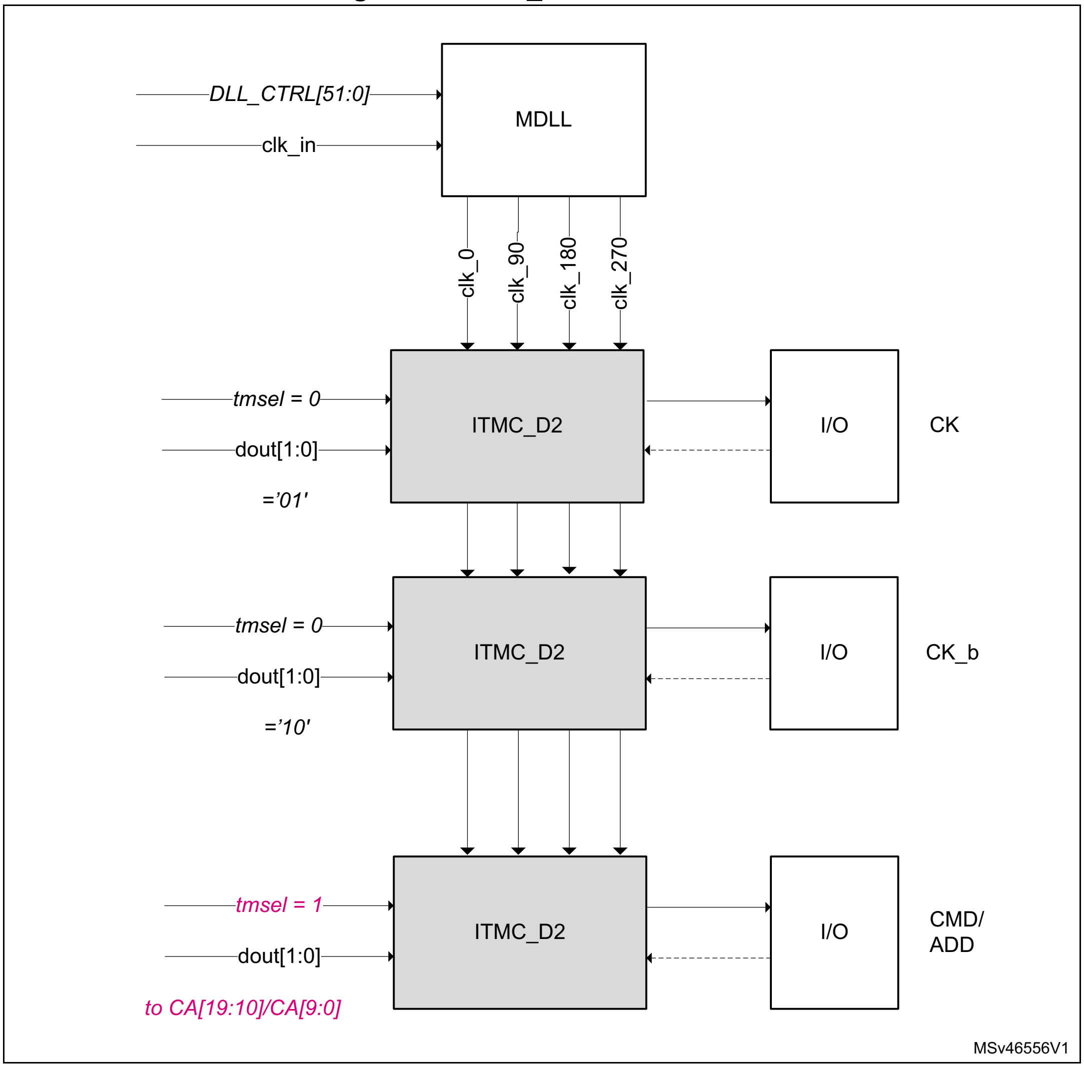
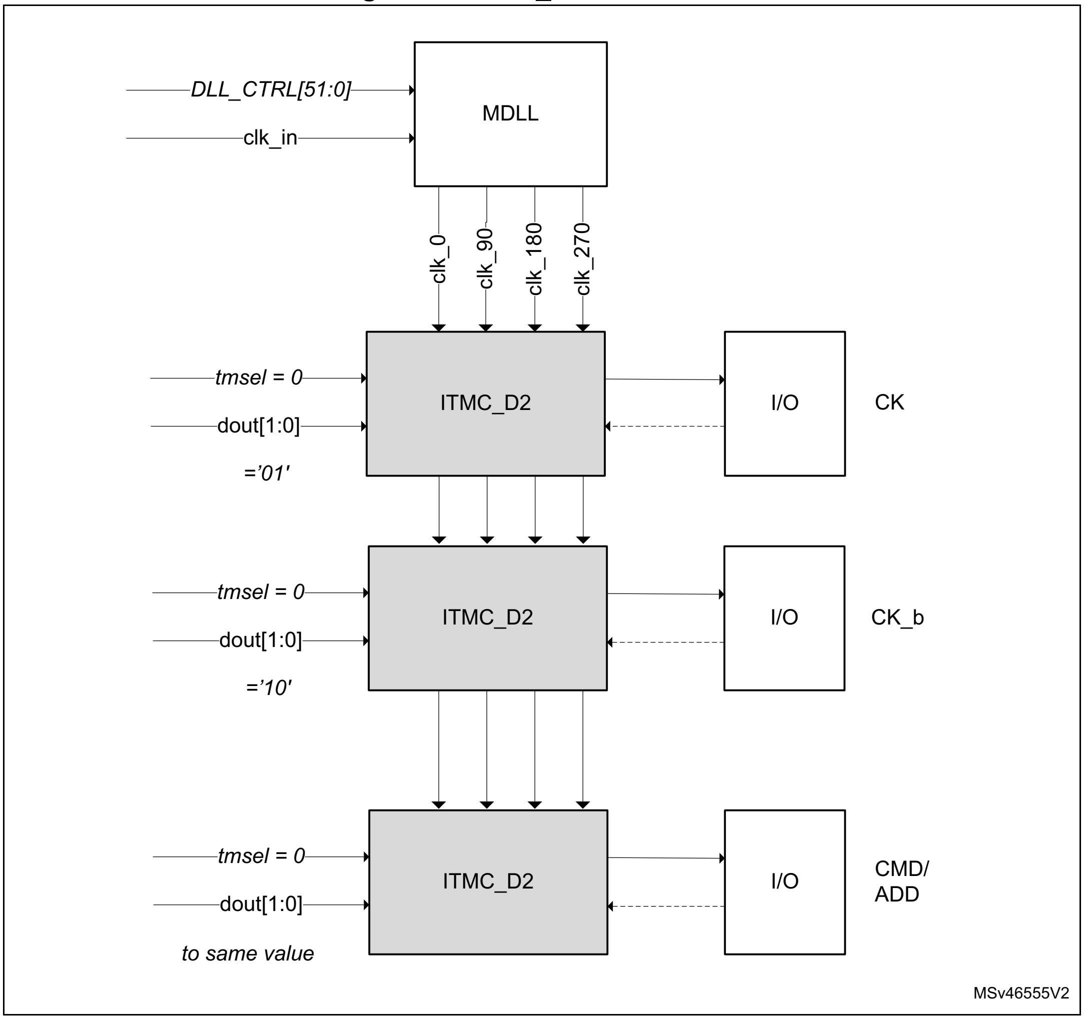


Figure 18. ITMC_D2 for DDR3/3L



The ITMS are used for DQ strobes (DQS/DQS#) with:

- SDR to DDR signaling conversion in output direction.
- Phase select positioning
- Fine step delay
- Drift monitoring support

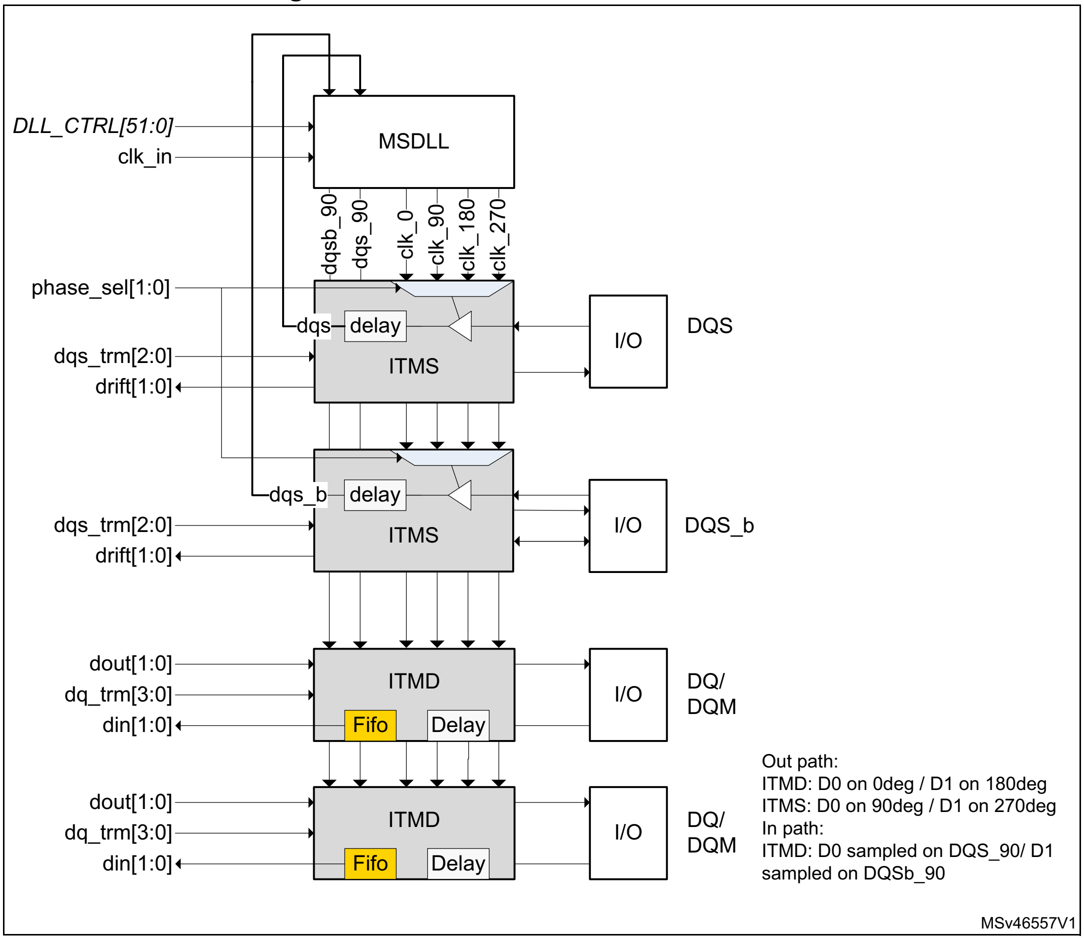
ITMS connections and timings are shown in *Figure 19*.

ITMD are used for data lanes (DQ[] and DQM) with:

- SDR to DDR signaling conversion in input and output directions.
- Sampling of incoming dq input on DQS and DQS# 90deg phases into a read FIFO
- Fine step sampling delay.

ITMD connections and timings are shown in *Figure 19*.

Figure 19. ITMS and ITMD for DQ/DQS lanes



7.4.4 SSTL IO's

SSTL IO key features:

- DDR3/DDR3L/LVCMOS operating modes
- Compatible with JEDEC standard for DDR3/DDR3L SDRAMs
- Programmable termination for DDR3/3L (ODT)
- Programmable output impedance
- PVT-compensated ODT and output impedance
- Driver and receiver power-down control
- Embedded boundary scan support logic
- PAD and internal loopback modes
- Core and I/O power down options supported
- Retention feature maintains I/O cell state during VDD power down

ZQ calibration cell:

The ZQ calibration cell provides independent calibration capability for the pull-up and pull-down ODT termination and for output impedances of the SSTL I/Os.

There are four sense blocks that provide independent sense capability for each impedance element:

– Pull-up termination impedance.

– Pull-down termination impedance.

– Pull-up output impedance.

– Pull-down output impedance.

These four elements are calibrated in series using the calibration select input ZCAL[1:0]. The calibration sequence is:

1. Output impedance pull-down.
2. Output impedance pull-up.
3. On-Die termination (ODT) pull-down.
4. ODT pull-up.

7.5 PUBL

The PUBL is intended to ease the control of the PHY features. It supports the PHY initialization, DQS gate training and programmable configuration control.

PUBL supports:

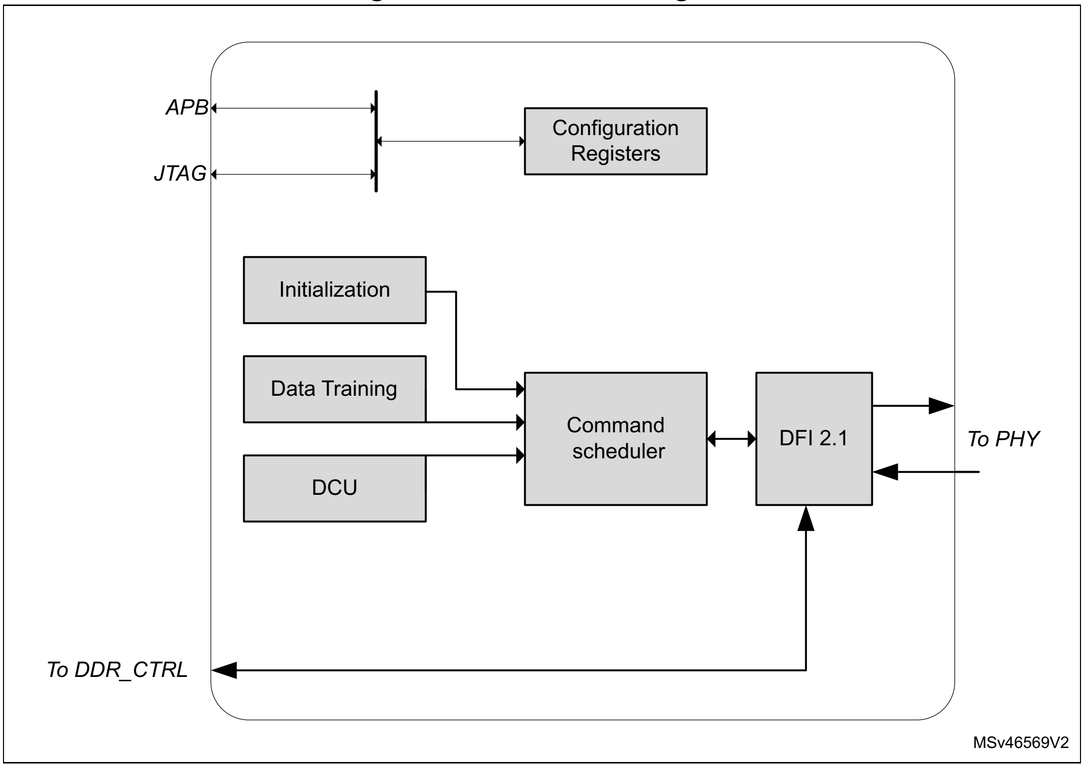
1. DDRPHY and SDRAM initialization sequences for all types of memories: DDR3/3L, LPDDR2 and LPDDR3.
2. A built-in DQS gate training (DQSTRN) to assist in programming the read DQS gate timing.
3. A DRAM Command Unit (DCU) to assist in programming the DRAM for timing calibration purposes.

The PUBL supports a DFI 2.1 interface with the DDRCTRL.

The PUBL includes configuration registers accessible via an APB interface. It also provides a JTAG interface access for test purposes.

A simplified block diagram of PUBL is shown in *Figure 20*.

Figure 20. PUBL block diagram



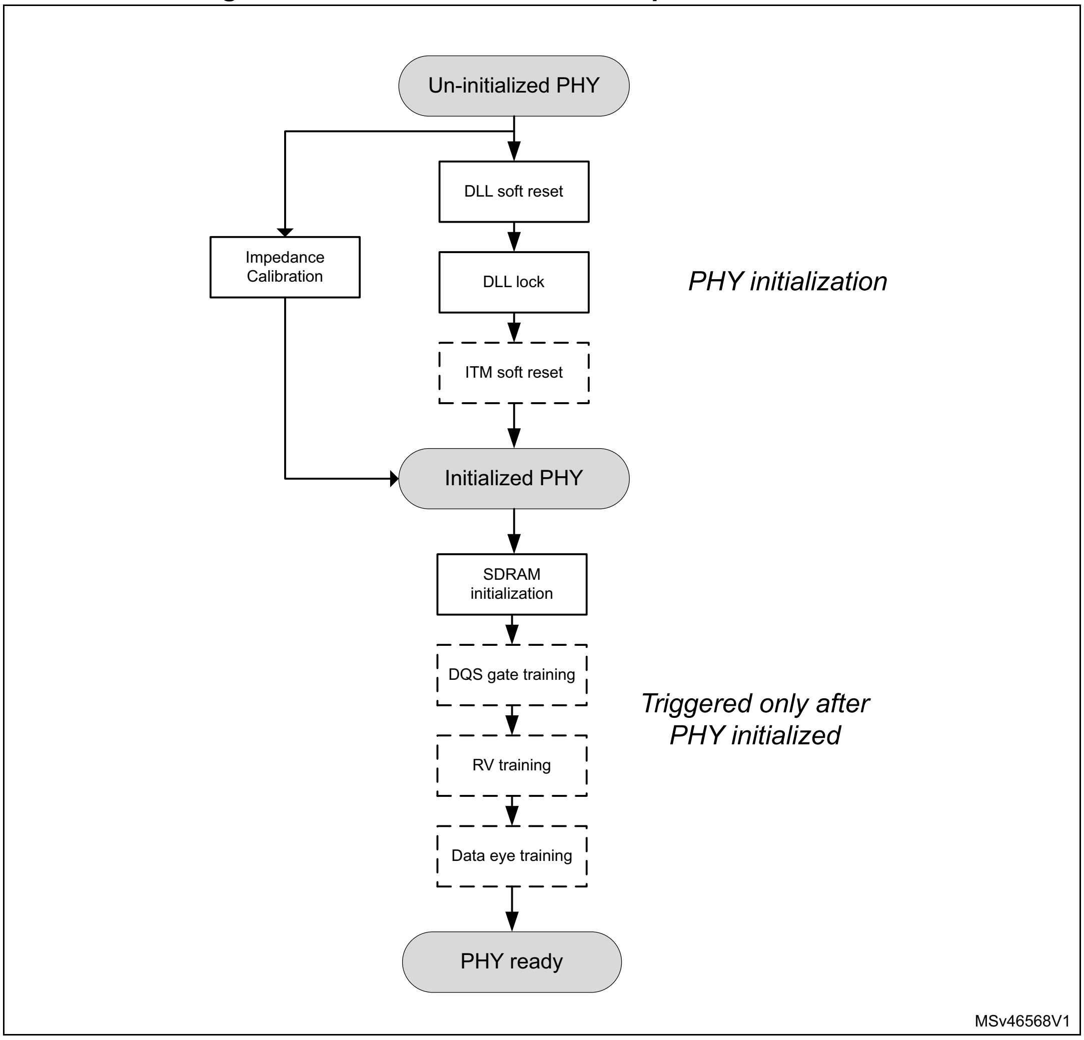
The following is a brief overview of the different design blocks and functionality of the PUBL:

- **Configuration registers**: includes all registers required to configure, control, train and self-test the PHY. The configuration block also contains registers required to initialize the DRAM.
- **Initialization**: used to automatically initialize the PHY, including locking of the DLL and calibration of I/O impedances. The initialization block also implements built-in initialization of all supported DRAMs.
- **Data training:** used to train the PHY for optimum timing margins. This includes finding the best positioning of the DQS gating window during reads, read valid generation and optimizing the read data eye.
- **DRAM command unit (DCU)**: allows the issuing of DRAM commands through the configuration port.
  This may be used to implement software DRAM initialization. It is also used for silicon debug and characterization of the PHY and the DDR interface.
- **Command scheduler**: multiplexes commands from internal PUBL blocks and the external memory controller and generates correct DRAM transactions in DFI 2.1 format. The commands from the external controller are passed to the DFI 2.1 block unmodified.
- **DFI 2.1 interface**: implements a DFI 2.1-compliant interface between the controller DDRCTRL and the PHY.

7.5.1 PHY and SDRAM initialization

Figure 21. shows a high-level diagram of the PHY initialization sequence.

Figure 21. DDRPHY initialization sequence



The initialization sequence has two phases. The first phase (shown in solid lines in *Figure 21*) happens automatically at reset and is used as follows:

1. Before and during configuration reset, the PHY is un-initialized and remains in this state until the reset is de-asserted.
2. At reset de-assertion, the PHY moves into the DLL initialization phase.
   This phase may be bypassed at any time by writing a '1' to bit LOCKBYP of the DDR PHY initialization register (DDRPHYC_PIR).
3. In parallel to DLL initialization, the impedance calibration phase also starts at reset de-assertion.
   This phase can also be bypassed by writing a '1' to the ZCALBYP bit of DDR PHY initialization register (DDRPHYC_PIR).
4. If the PHY initialization sequence was triggered by the user, a soft reset may optionally be selected to be issued to the ITMs.
   Initialization that is automatically triggered on reset does not issue a soft reset to the ITMs because the components have been already reset by the main reset.
5. Once the DLL initialization and impedance calibration phases are done and after the ITMs are reset, the PHY is initialized.

If these phases were bypassed, it is up to the user to perform them in software or trigger them at a later time before the PHY can be used.

The second phase of initialization starts after the PHY is initialized. Each step of this phase is triggered by the user (APB interface) or by DDR_CTRL:

1. The user or memory controller performs SDRAM initialization sequence (see “SDRAM Initialization”). The DRAM mode registers and necessary timing parameters must first be programmed into the PUBL **before** triggering SDRAM initialization.
2. After the SDRAM is initialized, the user or memory controller performs, or triggers the PUBL to perform, DQS gate training (see “Built-in DQS Gate Training” in *Section 7.5.6: DQS gate training (DQSTRN)*). The SDRAM must be initialized before triggering DQS gate training.
3. The user or memory controller performs, or triggers the PUBL to perform, read data valid training. PHY Initialization may be either Reset_triggered or PIR-triggered
4. The user or memory controller performs, or triggers the PUBL to perform, read data eye training.
   Note: the current version of the PUBL does not have the read eye training designed in.
5. The PHY is now ready for SDRAM read/write accesses.

The PUBL has a built-in SDRAM initialization routine that may be triggered by software or DDR_CTRL by writing to the PHY initialization register (DDRPHYC_PIR).

The initialization routine built into the PUBL is generic and does not require any knowledge of the type or configuration of external SDRAMs to be properly executed. The routine is designed with the relevant JEDEC specifications for the fastest and slowest SDRAMs supported by the PUBL to result in a universal initialization sequence. This generic sequence is applicable to DDR3 and LPDDR2 and DDR SDRAMs.

It is recommended to use the built-in PUBL routine to initialize the SDRAM. However, there may be cases such as during system debug when the built-in PUBL DRAM initialization is not triggered and DRAM initialization is performed by software or by the controller.

In these cases the system must first wait for the PHY to initialize, i.e. DLL locked and impedance calibration done, then it must write a '1' to the INIT bit of DDRPHYC_PIR with bit CTLDINIT of DDRPHYC_PIR set to '1' (for controller initialization) or '0' (for software or PUBL initialization) to inform the PUBL that DRAM initialization is done later, by software, the controller or by re-triggering on the PUBL.
The software or controller then executes the initialization sequence by sending relevant commands to the DRAM, respecting the various timing requirements of the initialization sequence.

The necessary configurations of the SDRAM must first be programmed into the PUBL before triggering SDRAM initialization.

SDRAM initialization must be triggered only after the PHY has been initialized.

All SDRAM initialization timing parameters described below are programmable in controller clock cycles in PTR1 and PTR2 registers.

The following sections describe the built in SDRAM initialization routines for different types of SDRAMs.

7.5.2 DDR3 initialization sequence

The initialization steps for DDR3 SDRAMs are as follows:

1. Optionally maintain RESET# low for a minimum of either 200 us (power-up initialization) or 100ns (power-on initialization).
   The PUBL drives RESET# low from the beginning of reset assertion and therefore this step may be skipped if enough time may already have expired to satisfy the RESET# low time.
2. After RESET# is de-asserted, wait a minimum of 500 us with CKE low.
3. Apply NOP and drive CKE high.
4. Wait a minimum of tXPR
   Caution: PUBL registers, SDRAM mode registers, and equivalent register fields inside the memory controller must be uniformly programmed. Mismatches between the register fields can cause transaction failures. Verify all register fields are consistently programmed before starting any SDRAM transaction.
5. Issue a load mode register 2 (MR2) command.
6. Issue a load mode register 3 (MR3) command.
7. Issue a load mode register (MR1) command (to set parameters and enable DLL).
8. Issue a load mode register (MR0) command to set parameters and reset DLL.
9. Issue ZQ calibration command.
10. Wait 512 SDRAM clock cycles for the DLL to lock (tDLLK) and ZQ calibration (tZQinit) to finish.
    This wait time is relative to Step 8, i.e. relative to when the DLL reset command was issued onto the SDRAM command bus.

7.5.3 LPDDR2 initialization sequence

The initialization steps for LPDDR2 SDRAMs are as follows:

1. Wait a minimum of 100 ns (tINIT1) with CKE driven low.
2. Apply NOP and set CKE high.
3. Wait a minimum of 200 us (tINIT3).
4. Issue a RESET command.
5. Wait a minimum of 1 us + 10 us (tINIT4 + tINIT5).
6. Issue a ZQ calibration command.
7. Wait a minimum of 1 us (tZQINIT).
8. Issue a Write mode register to MR1.
9. Issue a Write mode register to MR2
10. Issue a Write mode register to MR3

7.5.4 Initialization trigger and bypass

All initialization steps shown in *Figure 21* can be triggered using the PHY initialization register (DDRPHYC_PIR).
Writing a '1' to bit INIT of DDRPHYC_PIR register bit starts initialization, with the routines to be run selected by the DDRPHYC_PIR register bits. If multiple routines are selected, they are run in the order shown in *Figure 21*.

The completion of the routines is indicated in the PHY General Status Register (DDRPHYC_PGSR) with the corresponding done status bits.
The DDRPHYC_PGSR.IDONE bit indicates the overall completion of the initialization

sequence.
A status register bit is cleared (reset to '0') when the corresponding routine is re-triggered.

The de-assertion of reset automatically triggers the PUBL to perform **DLL initialization** with DLL locking and impedance calibration. Once the DLL has locked and impedance calibration has completed, the SDRAM initialization and DQS gating may be triggered or performed by software or memory controller.

Since the PUBL allows the selection of individual routines to be run when initialization is triggered using DDRPHYC_PIR register, the routines that automatically trigger on reset de-assertion have individual bypass capability.
This means that DLL locking and/or impedance calibration may be bypassed any time by writing a '1' to the corresponding bypass register bit in the DDRPHYC_PIR register. Once a routine is bypassed, it is internally registered as completed and the corresponding done status register bit is set in the PGSR register.

It is up to the user to re-trigger or perform the bypassed routines at a later time, before the PHY can be used.

Bit INITBYP of DDRPHYC_PIR register provides the option to bypass the whole initialization sequence

7.5.5 DATA training

After the DDRPHY and the external SDRAM have been successfully initialized, DDRPHY may be trained for optimum operating timing margins.
This includes:

- Training for DQS gating during reads (DQSTRN)
- Training for optimal Read Valid placement (RVTRN)

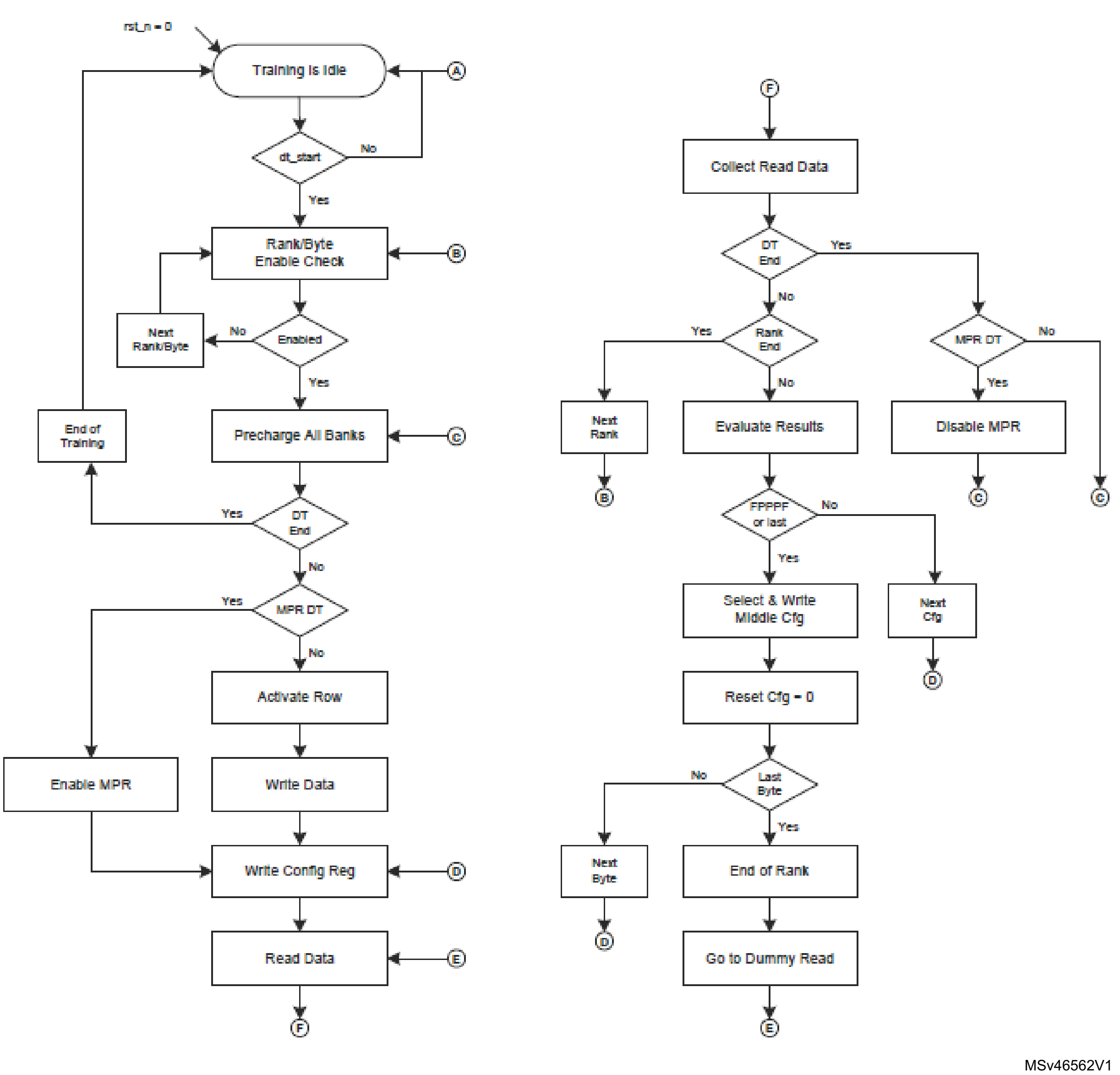
7.5.6 DQS gate training (DQSTRN)

Built-in DQS gate training:

Note: The built-in DQSTRN is mandatory and applicable only to DDR3/3L and may be using the MPR register.

The PUBL has a built-in DQS gate training routine that may be triggered by software or memory controller using the DDRPHYC_PIR register. *Figure 22* shows the detailed flow of the DQS gate training algorithm implemented in the PUB. This is followed by a description of the main phases of the training.

Figure 22. Buit-in DQS gate training flow



DQS gate training phases are as follows:

1. Data Train Idle: before and during reset, the data training state machine is in the idle state and remains in this state until the reset is de-asserted. The next configuration register values to be used for DQS gating are also reset to zero. These are the values that are written to DQS Gating System Latency (R0DGSL) and DQS Gating Phase Select (R0DGPS) fields of the DDRPHYC_DXnDQSTR register before each iteration of DQS gate training. Once reset is de-asserted, the FSM continues to remain in the idle state until data training is triggered. The FSM also remains in the idle state if data training is triggered with no ranks or bytes enabled. Otherwise the FSM goes to check if the next rank or byte is enabled.
2. Rank/byte enable check: this phase checks if the next rank/byte to be trained is enabled, otherwise it skips the rank/byte and goes to the next rank/byte. Bytes are

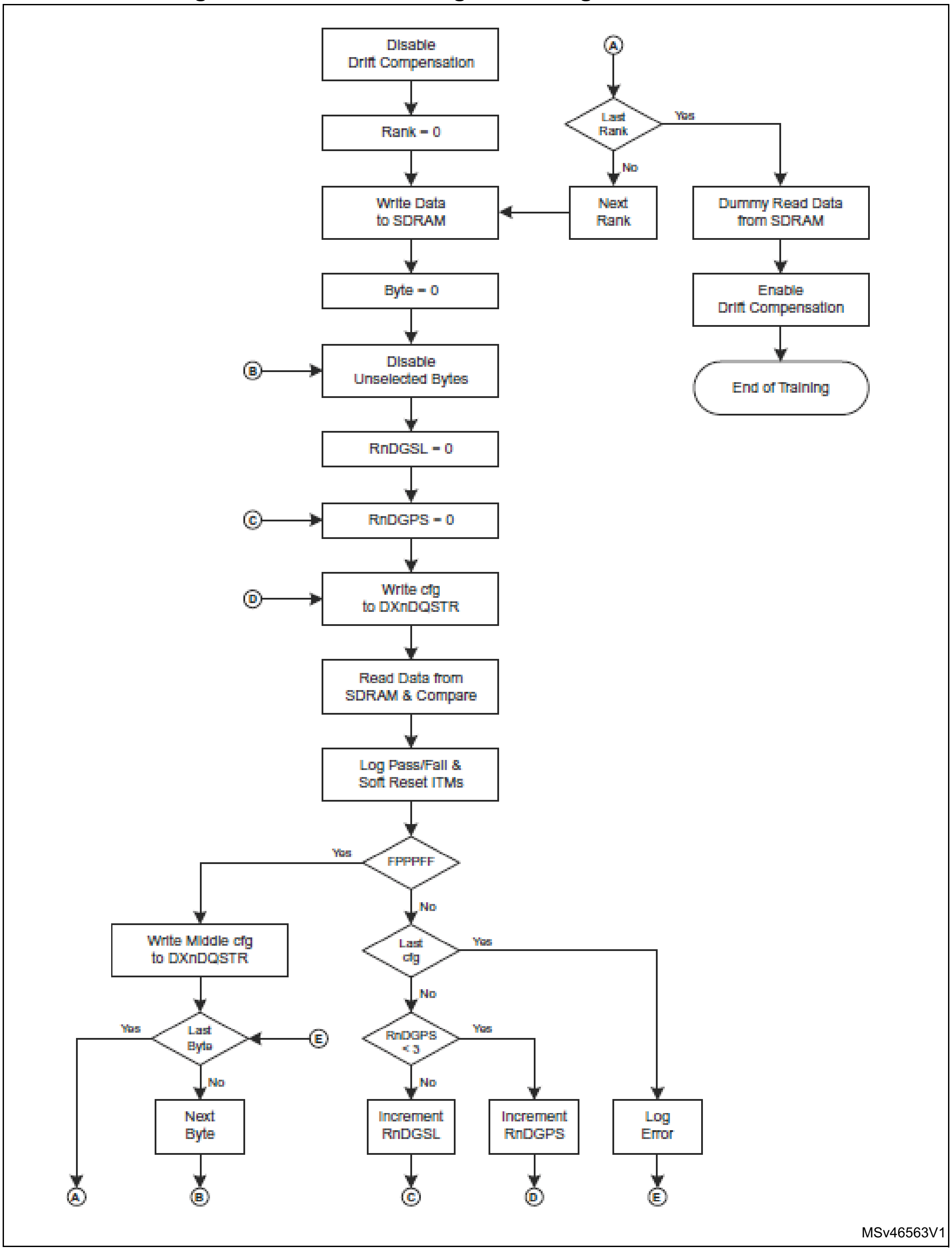
enabled/disabled using bit DXEN in DDRPHYC_DXnGCR register, and ranks are enabled/disabled using field RANKEN in DDRPHYC_DXnGCR register.

3. Precharge all banks: all banks are precharged before the beginning of training for each rank and at the end of the whole training. If the precharge is at the end of the training, the FSM goes to idle state, indicating the end of training. Otherwise it goes to prepare the DRAM for accesses, i.e. enable the MPR register if data training using MPR is selected, or activate the DRAM row to be used for write/read accesses.
4. Enable MPR: issues a load mode register command to the DRAM to enable the MPR register. This is only applicable for DDR3 DRAMs and only if the MPR bit is set in the DTAR register.
5. Activate: issues an activate command to the DRAM to open the bank row prior to writing data for training. The DRAM address used for training is user programmable in the DTAR register.
6. Write data: issues an 8-beat write to the DRAM to load data for training. The DRAM address and DRAM data used for training is user programmable in the DTAR, DTDTR0, DTDTR1 registers.
7. Write configuration register: once the data has been written to the DRAM (or MPR register has been enabled), the FSM writes the next configuration setting for the byte to the respective DQS gating register fields (DDRPHYC_R0DGSL and DDRPHYC_R0DGPS).
8. Read/collect data: this phase is executed either as a dummy read at the end of rank training or end of whole training (to reset the drift status registers in the PHY ITMs to the correct values), or it is executed as a normal read to evaluate if the DQS gate configuration is valid. After a dummy read, the FSM goes on to train the next rank or if it is the end of the whole training it goes to issue the last precharge or disable the MPR. After a read has been issued to the DRAM the FSM waits for all read data to come back. There is a time-out in case the read data does not come back. For a non-dummy read, the FSM goes on to evaluate the read data for pass/fail status.
9. Evaluate result: the read data is compared to the expected data and pass/fail status is generated and saved. If the status is such that a fail-pass-fail region (FPPPPF) has been found or if this is the last possible configuration (i.e. no passing region for this byte) the midpoint of the passing region is selected and written to (DDRPHYC_DXnDQSTR region. This signifies the end of training for the byte on the selected rank. If this is the last byte of the rank, the FSM goes on to train the next rank, otherwise it goes to the next byte of the rank. There is always a dummy read performed at the end of training all bytes of the rank.

Software DQS gate training

The DQS gate training may also be executed in software using DDRCTRL and/or the PUB DCU. *Figure 23* shows the DQS gate training software algorithm. This is followed by a description of the main phases of the training.

Figure 23. Software DQS gate training flow



The software DQS gate training phases are as follows:

1. Disable drift compensation by writing '0' to bit DFTCMP in DDRPHYC_DXnGCR register.
2. Start with rank 0, (DDRPHYC is supporting single rank only).
3. Execute a minimum of two writes to the SDRAM. Any type of data and any SDRAM address can be used for DQS gate training. It is however not recommended to use data

that is all zeroes since this may mask read data comparison. The data mask must be set to 0 to enable writing of all bytes. The number of writes must be chosen such that it results in a minimum of eight data beats at the SDRAM. This means at least two write commands when using SDRAM burst length of 4.

4. Start with byte 0 (i.e. byte 0 is selected for training).
5. Disable all the other bytes except the byte that has been selected for training. The bytes are enabled/disabled by writing 1/0 to bit DXEN in DDRPHYC_DXnGCR register. To avoid having to add programming and delay for a DLL re-lock when later re-enabling a disabled byte Lane, it is recommended to also set bit PDDISDX=0 in DDRPHYC_DXnGCR register before disabling any byte lanes. This prevents the byte Lane's DLL from going into bypass mode when a byte Lane is disabled with bit DXEN=0 in DDRPHYC_DXnGCR register.
6. Start with the selected byte DQS gating system latency (RnDGSL=0 in DXnDQSTR register).
7. Start with the selected byte DQS gating phase select (RnDGPS=0 in DXnDQSTR register).
8. Write the selected DQS gating configurations (bit R0DGSL and R0DGPS in register DXnDQSTR of the selected byte).
9. Execute reads from the SDRAM locations previously written. The number of reads must be equal to the number of writes used in Step 3. Compare the read data with the expected (written) data and log the pass/fail status as a sequence or history of flags for each trained RnDGSL/RnDGPS configuration (e.g. FFPPPPFF). A fail is either when there is a data mis-compare or when fewer data than expected is returned.

Note: A controller that is designed to always wait for the correct number of read data may need a time-out in case the trained configuration results in fewer data than expected. This is not as issue when using the PUB DCU because it does not wait for the expected number of reads; rather the read count status indicates if fewer reads were returned.

10. Once the read data has been compared and the pass/fail status logged, issue an ITM soft reset to clear the status of the read data logic in the PHY. This is important because the ITM read data FIFO pointers may be in the wrong state at the end of training an RnDGSL/RnDGPS configuration that resulted in wrong DQS gating window.
11. If two consecutive fails and some passes exist, then this is the end of the training for this rank byte. In this case, do the following:
    - Select the middle of the passes and write the values to the corresponding fields of DDRPHYC_DXnDQSTR register.

      - If this is not the last byte, then select the next byte and go to Step 5.

      - If this is the last byte but not the last rank, then select the next rank and go to Step 3.

      - If this is the last byte and the last rank, then go to Step 12 to do final clean-up before the end of the DQS gate training.
    - If the condition of two consecutive fails and some passes does not exist, then this signals that more RnDGSL/RnDGPS configurations need to be trained for this rank byte. If this is the case, do the following:

      - if R0DGPS is less than 3, then increment R0DGPS and go to Step 8

      - if R0DGPS is equal to 3 but R0DGSL is less than 7, then increment R0DGSL and go to Step 7

      - if R0DGPS is equal to 3 and R0DGSL is equal to 7, then log an error because this is a signal that something in the system is very wrong such that no passing configuration is possible for this rank byte. With such an error condition, you can either terminate the whole training to investigate the system or you can go to train the next byte.

12. Once the training of all ranks and all bytes is finished, issue one or more dummy reads to the same SDRAM locations. This flushes out the DQS drift compensation logic in the PHY and therefore avoid reporting any false drift events caused by previous DQS gating settings.
13. Once the dummy reads have completed, re-enable drift compensation by writing 1 to bit DFTCMP in DXnDQSTR register. This is the end of DQS gate training. Regular memory operations can now commence.

For details of how the PUBL DCU can be used to SDRAM accesses, please refer to *Section 7.5: PUBL*.

Note: It is not necessary to issue auto-refresh commands to the SDRAM during software DQS gate training if there is no data in the SDRAM other than the DQS gate training data. This is because the same locations are being read over and over within a relatively short period of time that should normally be less than the refresh timing requirement. However, if there is other data in the SDRAM and the training is taking longer than the required refresh timing, then refresh commands must be issued at the required intervals, interspersed relevantly with the training commands.

Read DQS gate training with low frequency LPDDR

When using LPDDR SDRAM, the built-in read DQS gate training is accurate for standard clock frequencies.

However, as the LPDDR tAC timing parameter becomes a smaller percentage of the clock period, fewer read DQS gate settings work. The read DQS gating works at low frequencies with LPDDR SDRAM, but this section provides additional guidance on how to correctly set up read DQS gating at low frequencies with LPDDR SDRAM

For all values of tAC >= (0.25*tCK), normal training can be used. When using slower speeds where tAC is less than 1 tCK, the number of data training passes may be less than ideal (as low as two) but the gate position is still satisfactory, even when aligned with rising edge of DQS. This is because there is enough margin before starting to see data fails (90 degrees of margin and at low speed is very large number).

The following lists the number of passing configurations for each of these settings and the gate position:

tAC ≥ 1.0*tCK

– Five passes, gate opening is positioned correctly 90degrees before rising edge of DQS

tAC ≥ 0.75*tCK < 1.0*tCK

– Four passes, gate opening is positioned correctly either aligned with rising edge of DQS or 90 degrees before rising edge of DQS

tAC ≥ 0.5*tCK < 0.75*tCK

– Three passes, gate opening is positioned correctly aligned with rising edge of DQS

tAC ≥ 0.25*tCK < 0.5*tCK

– Two passes, gate opening is positioned correctly aligned with rising edge of DQS

*Note: The exception to this guidance is when tAC < 0.25*tCK. In this case, no training should be performed and passive windowing should be used. Also, ensure the bits DGSL and DGPS are set to zero (DGPS is 1 by default.)*

7.5.7 Read valid training (RVTRN)

The PUBL has a built-in read valid training routine (RVTRN) that may be triggered by software or the memory controller using the DDRPHYC_PIR register.

For DDR3 RVTRN is not required

For LPDDR2/3 RVTRN is normally run at the same time as DQSTRN; it is adjusting the read valid system latency in extreme conditions with tDQSCK timings values.

7.6 PUBL timings and DFI interface

Note: This section is intended to show signals timing at the DDRPHYC interfaces and the relation between the signals at DFI and the SDRAM interface, with information about physical delays which are programmed into DDRCTRL.

7.6.1 Introduction

This section describes the timing for the interface between the controller and the PHY.

The read/write timing waveforms include four groups of signals:

1. DFI signals (dfi_*)
2. PHY clocks (ctl_clk, clk_0,clk_90, clk_180, and clk_270)
3. Controller-to-PHY interface signals (ctl_cmd, ctl_ds_oe, ctl_d_oe, ctl_d,ctl_qs_en, ctl_qs_dis, and ctl_qs_phase)
4. SDRAM signals (ck/ck_n, cmd, dqs/dqs_n, and dq).

only the important signals whose the timing needs to be controlled are shown, that are dfi_* and SDRAM signals (DDR_*).

The PHY clock group is not shown and ctl_clk is identical to dfi_clk.

From the controller-to-PHY interface signal group, only the ctl_qs_en signal is shown with the PHY passive DQS gating timing.

The read/write waveforms are given for typical PHY configurations and SDRAM latencies; the waveforms are shown for DDR3 mode.

The waveforms are mostly the same for active and the passive DQS gating, only the passive DQS gating timing is shown with the ctl_qs_en signal.

The timing waveforms shown are for write latency (WL) of 5 and read latency (RL) of 5. The timing of any latency, RWL, can be derived from these representative latencies by simply shifting the timing of (ctl_ds_oe, ctl_d_oe, ctl_d, ctl_qs_en, ctl_qs_dis, and ctl_qs_phase) relative to ctl_cmd by RWL controller clocks.

For DDR3, write latency (WL) is equal to CAS write latency (CWL) plus additive latency (AL), and read latency (RL) is equal to CAS latency (CL) plus additive latency (AL).

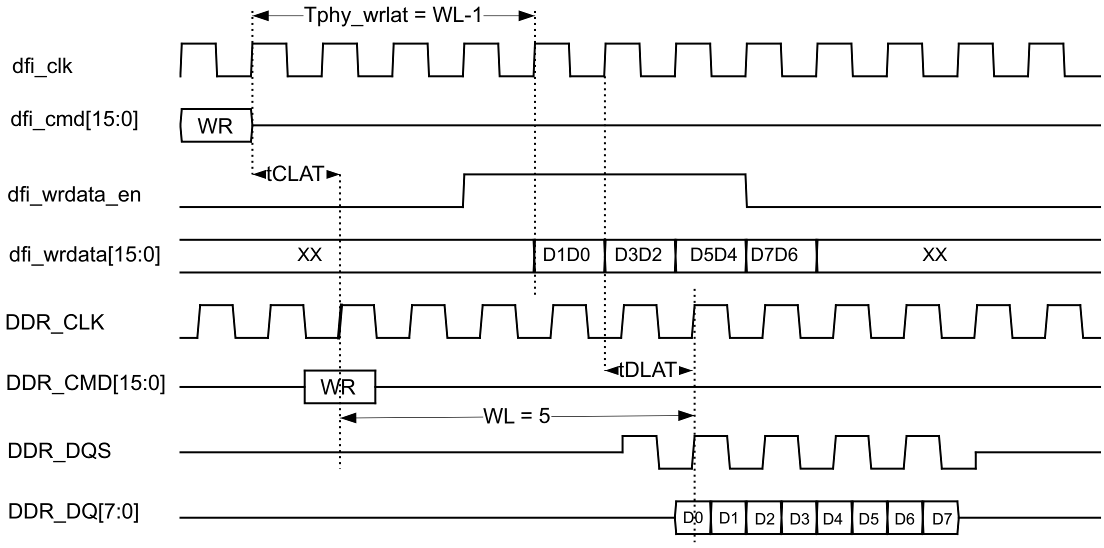
7.6.2 Write timing

Figure 24 shows the write timing for DDR3 mode with WL = 5 and BL = 8
Note that dfi_clk and ctl-clk are the same clocks, only dfi_clk is shown. The only variable in the DFI timing is the tphy_wrlat, which is the spacing between DFI write command and DFI write data enable (dfi_wrdata_en) signal.

As shown in *Table 25*, this spacing is equal to WL-1. The DFI controller always drives the write data (dfi_wrdata) one clock after it has driven the write data enable signal.
The timing of DFI write data mask (dfI_wrdata_mask) is the same as the timing of DFI write data (dfi_wrdata) and is therefore not shown in the figure. The DFI write command (dfi_cmd) is simply buffered to, and is therefore aligned with, the PHY controller command (ctl_cmd).

Figure 24 also shows the two PHY latency parameters for the command and data from the controller to the SDRAM. The command latency (tCLAT) is measured from the rising edge of the controller clock (ctl_clk) when the command is available to the PHY to the rising edge of SDRAM clock (DDR_CLK) when the command is output. Similarly, write data latency (tDLAT) is measured from the controller clock rising edge when data is available from the controller to the rising edge of data strobe (DDR_DQS) on which SDRAM data is available.

Figure 24. Write timing with WL = 5, BL = 8



Write timing: WL = 5, BL = 8

MSv46567V1

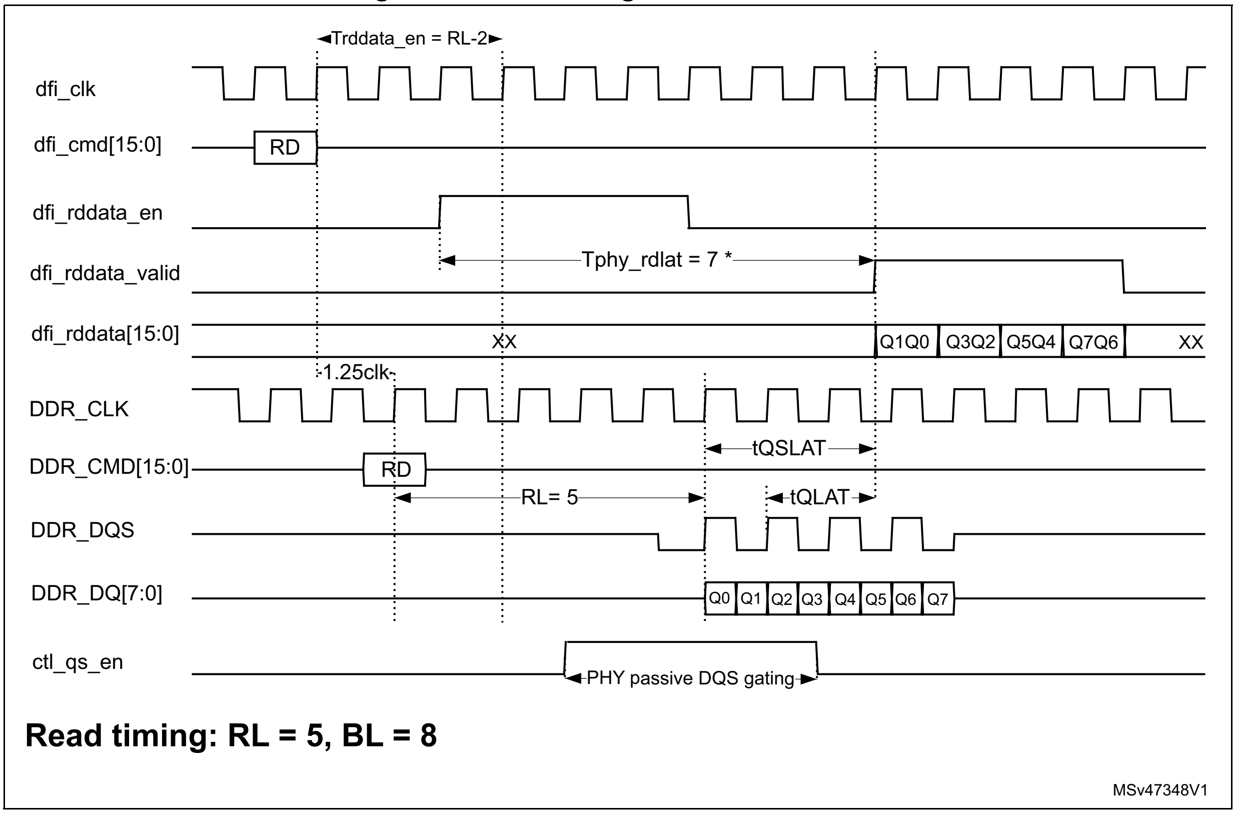
7.6.3 Read timing

Figure 25 shows the read timing for the DDR3 mode with WL = 5 and BL = 8.

Note that dfi_clk and ctl-clk are the same clocks, only dfi_clk is shown.
The only variable in the DFI timing is the trddata_en, which is the spacing between DFI read command and DFI read data enable (dfi_rddata_en) signal. As shown in *Figure 25*, this spacing is equal to RL-2.
The DFI read command (dfi_cmd) is simply buffered to, and is therefore aligned with, the PHY controller command (ctl_cmd).

The timing shown in *Figure 25* includes the timing of the read through the PHY read FIFO back to the controller. It also shows the two PHY latency parameters for the data from the SDRAM to the controller.
The data strobe latency (tQSLAT) is measured from the rising edge of the data strobe (DDR_DQS) that is coincident with the first data beat to the rising edge of the controller clock (ctl_clk same as dfi_clk) when data is sampled by the controller.
The read data latency (tQLAT) is measured from the rising edge of the data strobe (DDR_DQS) after two data beats are available to the rising edge of the controller clock (ctl_clk same as dfi_clk) when data is sampled by the controller.
The values of these parameters are described in *Table 24*. and *Table 25*.
Note that the read latencies have a variation of up to one controller clock cycle depending on the alignment of the read data strobe and the controller clock. This is because of the clock domain crossing between the read data strobe and the controller clock, which are assumed to be completely asynchronous to each other.

Figure 25. Read timing with WL = 5, BL = 8



7.6.4 DDRPHY read/write latency

Table 24: PHY latency shows the latency of commands and data through the PHY.

Refer to *Write timing on page 367* and *Read timing on page 368* for definitions and illustration of these parameters.

Table 24. PHY latency

SYMBOL	Description	Latency	Units	Note
t_{CLAT}	Command Latency	1.25	tCLK	-
t_{DLAT}	Write data Latency	1.25	tCLK	-
t_{QLAT}	Read data Latency	1-2	tCLK	1
t_{QSLAT}	Read data strobe Latency	2-3	tCLK	1

Note: The PHY Latency depends on the alignment of the read data strobe to the controller clock.

The PHY read latency parameter tphy_rdlat, i.e. the number of DFI clock cycles from the assertion of dfi_rddata_en signal to the assertion of dfi_rddata_valid is defined as:

tphy_rdlat = 3 + tQSLAT + 1 + x + y

With the no-bubbles feature enabled, the latency is:

tphy_rdlat = 3 + tQSLAT + 1 + x + y + 2

With the fixed-latency feature enabled, the latency is:

tphy_rdlat = 12

Where:

x = propagation delay of the command from the PHY macro outputs to the SDRAM

y = propagation delay of the read data from the SDRAM to the PHY macro inputs

gdqs_rsl = set of DQS gating system latency values for all ranks and all byte-lanes

The term (+3) and the term (+1) of the above equations are internal DFI logic latencies. The terms x and y include, among other things, the routing between the PHY and the I/Os, the I/O propagation delays, package delays, board delays, and other delays associated with the SDRAM that are not accounted for in the SDRAM read latency (RL) parameter.

The total read latency from the DFI read command to the DFI read data valid is

trddata_en + tphy_rdlat.

where trddata_en is described in *Table 25*.

7.6.5 DFI 2.1 overview

The DFI is an interface protocol that defines the connectivity between a DDR memory controller (DDRCTRL) and a DDR physical interface (DDRPHY) for DDR memory devices (here: DDR3 and LPDDR2).

The protocol defines the signals, signal relationships, and timing parameters required to transfer control information and data to and from the DRAM devices over the DFI.

The DDRPHYC is supporting protocol version 2.1:

Interface signal groups

DFI is subdivided into the following interface groups:

- Control interface
- Write data interface
- Read data interface
- Update interface
- Status interface
- Training interface
- Low power control interface

DFI timing parameters

The DFI defines several timing parameters with fixed values and specified as a number of DFI clock cycles. *Table 25* describes the DFI parameters applicable to DDRPHYC.

The DFI timings parameters are relevant for DDRCTL which needs to programmed accordingly.

Table 25. DFI timing parameters

Name	Description	SDR mode	Note
t_{ctrl_delay}	Specifies the number of DFI clock cycles after an assertion or deassertion of the DFI control signals that the control signals at the PHY-DRAM interface reflect the assertion or de-assertion. If the DFI clock and the memory clock are not phase-aligned, this timing parameter should be rounded up to the next integer value.	2	(1)
t_{phy_wrdata}	Specifies the number of DFI clock cycles between when the dfi_wrdata_en signal is asserted to when the associated write data is driven on the dfi_wrdata signal.	1	-
t_{phy_wrlat}	Specifies the number of DFI clocks between when a write command is sent on the DFI control interface and when the dfi_wrdata_en signal is asserted Note: In LPDDR2 mode, the following settings must be used: WL = MR2[3:0].WL + 1 (to account for tDQSS)	WL-1	(2)(3)(4)
t_{rddata_en}	Specifies the time from the assertion of a read command on the DFI to the assertion of the dfi_rddata_en signal. Note: In LPDDR2 mode, the following settings must be used: RL = MR2[3:0].RL + tDQSCK	RL-2	(2)(3)
t_{phy_rdlat}	Specifies the maximum number of cycles allowed from the assertion of the dfi_rddata_en signal to the assertion of the dfi_rddata_valid signal.	15	(5)
$t_{dram\_ck\_disable}$	Specifies the number of clocks from the assertion of the dfi_dram_clk_disable signal on the DFI until the clock to the DRAM memory devices, at the PHY-DRAM boundary, maintains a low value	2	-
$t_{dram\_ck\_enable}$	Specifies the number of clocks from the de-assertion of the dfi_dram_clk_disable signal on the DFI until the first valid rising edge of the clock to the DRAM devices at the PHYDRAM boundary.	2	-
t_{phyupd_type0}	Specifies the maximum number of DFI clocks that the dfi_phyupd_req signal may remain asserted after the assertion of the dfi_phyupd_ack signal for dfi_phyupd_type = 0.	1	(6)
t_{phyupd_resp}	Specifies the maximum number of DFI clock cycles after the assertion of the dfi_phyupd_req signal to the assertion of the dfi_phyupd_ack signal.	64	
$t_{phy\_paritylat}$	Specifies the maximum number of DFI clock cycles between when the dfi_parity_in signal is driven and when the associated dfi_parity_error is returned.	PL+6	(7)

1. Actual latencies are 1.25 but rounded up to the next integer as per DFI specification.
2. For LPDDR2 device usage the WL is the SDRAM write latency (CAS write latency + additive latency + 1) to ensure that the LPDDR2 tDQSS is accounted for. For all other DDR device usage the WL is the SDRAM write latency (CAS write latency + additive latency). If an external controller wants to run the PHY in loopback mode, i.e. if LBMODE is set, then tphy_wrlat must be set to WL-2.
3. In LPDDR2 mode, WL = MR2.WL + 1 and RL = MR2.RL + tDQSCK.
4. RL is the SDRAM read latency (CAS latency + additive latency).
5. The number includes the latency through the PHY (i.e. command latency plus read latency), plus maximum possible system latency that could be trained by the PUB.
6. This corresponds to the maximum time required for updating ITMs.
7. PL is the DIMM parity latency. it is N/A with current configuration.

7.7 PUBL registers

7.7.1 DDRPHYC revision ID register (DDRPHYC_RIDR)

Address offset: 0x000

Reset value: 0x0020 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRID[7:0]								PHYMJR[3:0]				PHYMDR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYMNR[3:0]				PUBMJR[3:0]				PUBMDR[3:0]				PUBMNR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **UDRID[7:0]**: User-defined rev ID

Bits 23:20 **PHYMJR[3:0]**: PHY maj rev

Bits 19:16 **PHYMDR[3:0]**: PHY moderate rev

Bits 15:12 **PHYMNR[3:0]**: PHY minor rev

Bits 11:8 **PUBMJR[3:0]**: PUB maj rev

Bits 7:4 **PUBMDR[3:0]**: PUB moderate rev

Bits 3:0 **PUBMNR[3:0]**: PUB minor rev

7.7.2 DDRPHYC PHY initialization register (DDRPHYC_PIR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INITBYP	ZCALBYP	LOCKBYP	CLRSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTLDINIT	DLLBYP	ICPC
w	w	w	w										w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVTRN	QSTRN	DRAMINIT	DRAMRST	ITMSRST	ZCAL	DLLLOCK	DLLSRST	INIT
							w	w	w	w	w	w	w	w	w

Bit 31 **INITBYP**: Initialization bypass
Bypasses or stops, if set, all initialization routines currently running, including PHY initialization, DRAM initialization, and PHY training.
Initialization may be triggered manually using INIT and the other relevant bits of the DDRPHYC_PIR register. This bit is self-clearing.

Bit 30 **ZCALBYP**: zcal bypass
Bypasses or stops, if set, impedance calibration of all ZQ control blocks that automatically triggers after reset.
Impedance calibration may be triggered manually using INIT and ZCAL bits of the DDRPHYC_PIR register. This bit is self-clearing.

Bit 29 **LOCKBYP**: DLL lock bypass
Bypasses or stops, if set, the waiting of DLLs to lock. DLL lock wait is automatically triggers after reset.
DLL lock wait may be triggered manually using INIT and DLLLOCK bits of the DDRPHYC_PIR register. This bit is self-clearing.

Bit 28 **CLRSR**: clear status register
Writing 1 to this bits does the following:
- Auto-clears itself. This means a following read to bit CLRSR returns 0
- Clears the DXnGSR0 bits for DTDONE, DTERR, DTIERR
- Clears bit DFTERR in DDRPHYC_PGSR register and bit DFTERR in DDRPHYC_DXnGSR1 register

This bit is primarily for debug purposes and is typically **not** needed during normal functional operation.
It can be used when bit IDONE=1 in PGSR register, to manually clear the PGSR status bits, however, the PGSR status bits (except for the DFTRR and TQ bits) are automatically cleared by starting a new init process.
The bit can also be used to manually clear the DXnGSR status bits, however, starting a new data training process automatically clears the DXnGSR status bits.

Bits 27:19 Reserved, must be kept at reset value.

Bit 18 **CTLDINIT**: Controller DRAM initialization
Indicates if set that DRAM initialization is performed by the controller. Otherwise if not set it indicates that DRAM initialization is performed using the built-in initialization sequence or using software through the configuration port.

Bit 17 **DLLBYP**: DLL bypass
A setting of 1 on this bit puts all PHY DLLs in bypass mode. A bypassed DLL is also powered down (disabled).

Bit 16 **ICPC**: Initialization complete pin configuration
Specifies how the DFI 2.1 initialization complete output pin should be used to indicate the status of initialization.
0: Asserted after PHY initialization (DLL locking and impedance calibration) is complete.
1: Asserted after PHY initialization is complete and the triggered PUBL initialization (DRAM initialization, data training, or initialization trigger with no selected initialization) is complete.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **RVTRN**:

it is expected that RVTRN and DQSTRN are run with LPDDR2/3; RVTRN is not required for DDR3

If RVTRN=1 and bit QSTRN=0, when bit INIT=1 (triggering the init process) the read DQS gate training algorithm still runs (as if bit QSTRN in DDRPHYC_PIR register was actually set to 1).

If it is necessary to run only RV training stand-alone, with no read DQS gate training, you can prevent the read DQS gate training from running by setting bit LBGDQS=1 in DDRPHYC_PGCR register.

Note: RV training cannot use multi-purpose register (MPR) and must use the user data programmed in DTDR0-1.

Bit 7 **QSTRN**: Read DQS training

Executes a PUBL training routine to determine the optimum position of the read data DQS strobe for maximum system timing margins.

Note: Read DQS training cannot use multi-purpose register (MPR) and must use the user data programmed in DTDR0-1.

Bit 6 **DRAMINIT**: DRAM initialization

Executes the DRAM initialization sequence

Bit 5 **DRAMRST**: DRAM reset (DDR3 only)

Issues a reset to the DRAM (by driving the DRAM reset pin low) and wait 200us.

This can be triggered in isolation or with the full DRAM initialization (DRAMINIT). For the latter case, the reset is issued and 200us is waited before starting the full initialization sequence.

Bit 4 **ITMSRST**: ITM reset

Soft resets the interface timing modules for the data and data strobes, i.e., it asserts the ITM soft reset (srstb) signal.

Bit 3 **ZCAL**: Impedance calibration (Driver and ODT)

Performs PHY impedance calibration.

Bit 2 **DLLLOCK**: DLL lock

Waits for the PHY DLLs to lock.

Bit 1 **DLLSRST**: DLL soft reset

Soft resets all PHY DLLs by driving the DLL soft reset pin.

Note: - Requires that ctl_clk be toggling for the DLL soft reset signal to be output from the DLL. If ctl_clk (AC DLL's cclk_0) is not guaranteed to be toggling, it is recommended to use the manual DLL soft reset bit DLLSRST of DDRPHYC_ACDLLCR, and not bit DLLSRST of DDRPHYC_PIR.

- Ensure that the minimum requirements for DLL bypass and DLL reset are observed while asserting this bit.

Bit 0 **INIT**: Initialization trigger

A write of '1' to this bit triggers the DDR system initialization, including PHY initialization, DRAM initialization, and PHY training.

The exact initialization steps to be executes are specified in bits **1 to 6** of this register. A bit setting of 1 means the step is executed as part of the initialization sequence, while a setting of ‘0’ means the step is bypassed.

The initialization trigger bit is self-clearing.

7.7.3 DDRPHYC PHY global control register (DDRPHYC_PGCR)

Address offset: 0x008

Reset value: 0x0184 2E04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBMODE	LBGDQS	LBDQSS	RFSHDT[3:0]				PDDISDX	ZKSEL[1:0]		RANKEN[3:0]				IODDRM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IOLB	CKINV	CKDV[1:0]		CKEN[2:0]			DTOSEL[3:0]				DFTLMT[1:0]		DFTCMP	DQSCFG	ITMDMD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LBMODE**: Loop back mode

Indicates if set that the PHY/PUB is in loopback mode

Bit 30 **LBGDQS**: Loop back DQS gating

Selects the DQS gating mode that should be used when the PHY is in loopback mode. Valid values are:

0: DQS gate training is triggered on the PUB

1: DQS gate is set manually using software

Bit 29 **LBDQSS**: Loop back DQS shift

Selects how the read DQS is shifted during loopback to ensure that the read DQS is centered into the read data eye. Valid values are:

0: PUB sets the read DQS delay to 0; DQS is already shifted 90 degrees by write path

1: The read DQS shift is set manually through software.

Bits 28:25 **RFSHDT[3:0]**: Refresh during training

A non-zero value specifies that a burst of refreshes equal to the number specified in this field should be sent to the SDRAM after training each rank except the last rank.

Bit 24 **PDDISDX**: Power down disabled byte

Indicates if set that the DLL and I/Os of a disabled byte should be powered down.

Bits 23:22 **ZKSEL[1:0]**: Impedance clock divider selection

Selects the divide ratio for the clock used by the impedance control logic. The source clock for

the divider is the configuration port clock signal (pclk), depending on which configuration port type used (see “Impedance Calibration” on page tbc).

Valid values are:

00: Divide by 2

01: Divide by 8

10: Divide by 32

11: Divide by 64

Bits 21:18 **RANKEN[3:0]**: Rank enable

Specifies the ranks that are enabled for data-training.

only Bit 0 is used and controls rank 0.

Setting the bit to ‘1’ enables the rank, and setting it to ‘0’ disables the rank.

Bits 17:16 **IODDRM[1:0]**: I/O DDR mode

Selects the DDR mode for the I/Os.

Bit 15 **IOLB**: I/O loop back select

Selects where inside the I/O the loop-back of signals happens. Not applicable to D3A I/Os. Valid values are:

0: Loopback is after output buffer; output enable must be asserted

1: Loopback is before output buffer; output enable is don't care

Bit 14 **CKINV**: CK invert

Specifies if set that CK/CK# should be inverted. Otherwise CK/CK# toggles with normal polarity.

Bits 13:12 **CKDV[1:0]**: CK disable value

Specifies the static value that should be driven on CK/CK# pair(s) when the pair(s) is disabled. CKDV[0] specifies the value for CK and CKDV[1] specifies the value for CK#.

Bits 11:9 **CKEN[2:0]**: CK enable

Controls whether the CK going to the SDRAM is enabled (toggling) or

disabled (static value defined by CKDV). One bit for each of the three CK pairs.

Bits 8:5 **DTOSEL[3:0]**: Digital test output select

Digital Test Output Select: Selects the PHY digital test output that should be driven onto PHY digital test output (DDR_DTO0) pin: Valid values are:

0000: DATX8 0 DLL digital test output

0001: DATX8 1 DLL digital test output

0010: DATX8 2 DLL digital test output

0011: DATX8 3 DLL digital test output

0100: DATX8 4 DLL digital test output

0101: DATX8 5 DLL digital test output

0110: DATX8 6 DLL digital test output

0111: DATX8 7 DLL digital test output

1000: DATX8 8 DLL digital test output

1001: AC DLL digital test output

Other: Reserved

Bits 4:3 **DFTLMT[1:0]**: DQS drift limit

Specifies the expected limit of drift on read data strobes. A drift of this value or greater is reported as a drift error through the host port error flag. Valid values are:

00: No limit (no error reported)

01: 90° drift

10: 180° drift

11: 270° or more drift

Note: Although reported through the error flag, this is not an error requiring any action. It is simply an indicator that the drift is greater than expected.

Bit 2 **DFTCMP**: DQS drift compensation

Enables or disables DQS drift compensation. Valid values are:
0: Disables data strobe drift compensation
1: Enables data strobe drift compensation
By default, drift compensation is enabled.

Note: Drift compensation is not supported under any of the following situations:
- LPDDR2/3 (bit DDRMD set to LPDDR2 in DDRPHYC_DCR register)
- Burst length 2 (bit BL in DDRPHYC_MR0 register set to burst length of 2)
- Read DQS gating using passive windowing (bit DQSCFG in DDRPHYC_PGCR register set to passive windowing)
Drift compensation must be set to disabled if any of the above are set.

Bit 1 **DQSCFG**: DQS gating configuration

Selects one of the two DQS gating schemes:
0: DQS gating is shut off using the rising edge of DQS_b (active windowing mode)
1: DQS gating blankets the whole burst (passive windowing mode).

Note: Passive windowing must be used for LPDDR2/3.

Bit 0 **ITMDMD**: ITM DDR mode

Selects whether ITMS uses DQS and DQS# or it only uses DQS.
Valid values are:
0: ITMS uses DQS and DQS#
1: ITMS uses DQS only

Note: The correct setting is always DQS and DQS#.

7.7.4 DDRPHYC PHY global status register (DDRPHYC_PGSR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	RVEIRR	RVERR	DFTERR	DTIERR	DTERR	DTDONE	DIDONE	ZCDDONE	DLDONE	IDONE
						r	r	r	r	r	r	r	r	r	r

Bit 31 **TQ**: Temperature output (LPDDR only) N/A

Bits 30:10 Reserved, must be kept at reset value.

Bit 9 **RVEIRR**: Read valid training intermittent error

If set, indicates that there was an intermittent error during read valid training, such as a pass was followed by a fail then followed by another pass.

Bit 8 **RVERR**: Read valid training error

If set, indicates that a valid read valid placement could not be found during read valid training.

Bit 7 **DFTERR**: DQS drift error

If set, indicates that at least one of the read data strobes has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR).

Bit 6 **DTIERR**: DQS gate training intermittent error
If set, indicates that there was an intermittent error during DQS gate training, such as a pass was followed by a fail then followed by another pass.

Bit 5 **DTERR**: DQS gate training error
If set, indicates that a valid DQS gating window could not be found during DQS gate training.

Bit 4 **DTDONE**: Data training done
Indicates, if set, that the PHY has finished doing data training.

Bit 3 **DIDONE**: DRAM initialization done
Indicates if set that DRAM initialization has completed.

Bit 2 **ZCDDONE**: zcal done
Indicates if set that impedance calibration has completed.

Bit 1 **DLDONE**: DLL lock done
Indicates if set that DLL locking has completed.

Bit 0 **IDONE**: Initialization done
Indicates if set that the DDR system initialization has completed.
This bit is set after all the selected initialization routines in DDRPHYC_PIR register have completed.

7.7.5 DDRPHYC DDR DLL global control register (DDRPHYC_DLLGCR)

Address offset: 0x010

Reset value: 0x0373 7000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLLRSVD2[1:0]		LOCKDET	FDTRMSL[1:0]		SBIAS5_3[2:0]			BPS200	SBIAS2_0[2:0]			MBIAS[7:4]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBIAS[3:0]				TESTSW	ATC[1:0]		DTC[2:0]			TESTEN	IPUMP[2:0]			DRES[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DLLRSVD2[1:0]**:
These bit are connected to the DLL control bus and reserved for future use.

Bit 29 **LOCKDET**: Master lock detect enable
Note: This field is only valid for a few processes. For all other processes this field is reserved.

Bits 28:27 **FDTRMSL[1:0]**: Slave bypass fixed delay trim
00: Nominal delay
01: Nominal delay – 10%
10: Nominal delay + 10%
11: Nominal delay + 20%

Bits 26:24 **SBIAS5_3[2:0]**: Slave bias trim
Used to trim the bias for the slave DLL.

Bit 23 **BPS200**: Bypass mode frequency range
0: 0 to 100 MHz
1: 0 to 200 MHz

Bits 22:20 **SBIAS2_0[2:0]**: Slave bias trim
Used to trim the bias for the slave DLL.

Bits 19:12 **MBIAS[7:0]**: Master bias trim
Used to trim the bias for the master DLL.

Bit 11 **TESTSW**: Test switch
Selects the test signals of either the master DLL, set to 0, or the slave DLL, set to 1.

Bits 10:9 **ATC[1:0]**: Analog test control
Selects the analog signal to be output on the DLL analog test output (DDR_ATO) when TESTEN is high (Output is Vss when TESTEN is low).
The test output either comes from the master DLL or the slave DLL, depending on the setting of the test switch (TESTSW).
Both master DLL and slave DLL output similar analog test signals. Valid settings for analog test control are:
00: Replica bias output for PMOS (Vbp)
01: Replica bias output for NMOS (Vbn)
10: Filter output (Vc)
11: V_{DDCORE}

Bits 8:6 **DTC[2:0]**: Digital test control
Selects the digital signal to be output on the DLL digital test output (DTO1 pin) when TESTEN is high (Output is '0' when TESTEN is low). The DTO0 pin output masters the DLL reference clock (clk_0).
Valid settings for **master DLL** (such as, when TESTSW = '0'):
- 000: 0 output clock (clk_0)
- 001: 90 output clock (clk_90)
- 010: 180 output clock (clk_180)
- 011: 270 output clock (clk_270)
- 100: 360 internal clock (clk_360_int)
- 101: Speed-up pulse (spdup)
- 110: Slow-down pulse (slwdn)
- 111: 0 MCTL logic clock (cclk_0)

Valid settings for **slave DLL** (such as when TESTSW = '1'):
- 000: Input DQS strobe (dqs)
- 001: Input clock reference (clk_90_in)
- 010: Internal feedback clock (clk_0_out)
- 011: 90 output DQS# strobe (dqsb_90)
- 100: 90 output DQS strobe (dqs_90)
- 101: Speed-up pulse (spdup)
- 110: Slow-down pulse (slwdn)
- 111: Auto-lock enable signal

Bit 5 **TESTEN**: Test enable
Enables digital and analog test outputs selected by DTC and ATC respectively.

Bits 4:2 **IPUMP[2:0]**: Charge pump current trim
000: maximum current
111: minimum current

Bits 1:0 **DRES[1:0]**: Trim reference current versus resistor value variation
00: Rnom
01: Rnom - 20%
1x: Rnom + 20%
Note: For a few processes this field is reserved and is not used by the DLL.

7.7.6 DDRPHYC AC DLL control register (DDRPHYC_ACDLLCR)

Address offset: 0x014

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTEN	Res.	Res.
rw	rw												rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	MFWDLY[2:0]			MFBDLY[2:0]			Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw	rw	rw						

Bit 31 **DLLDIS**: DLL disable
A disabled DLL is bypassed. By default DLL is enabled.
Note: DLL bypass must only be changed when DDRPHYC clock (dphy_ker_ck) is inactive.

Bit 30 **DLLSRST**: DLL soft reset
Soft resets the AC DLL by driving the DLL soft reset pin.
Note: DLL reset must only be changed to 0 when DDRPHYC clock (dphy_ker_ck) is inactive.

Bits 29:19 Reserved, must be kept at reset value.

Bit 18 **ATESTEN**: Analog test enable
Enables the analog test signal to be output on the DLL analog test output (ATO pin). The DLL analog test output is tri-stated when this bit is '0'.

Bits 17:12 Reserved, must be kept at reset value.

Bits 11:9 **MFWDLY[2:0]**: Master DLL feed-forward delay trim
Used to trim the delay in the master DLL feed-forward path:
000: minimum delay
111: maximum delay

Bits 8:6 **MFBDLY[2:0]**: Master DLL feed-back delay trim
Used to trim the delay in the master DLL feed-back path:
000: minimum delay
111: maximum delay

Bits 5:0 Reserved, must be kept at reset value.

7.7.7 DDRPHYC PT register 0 (DDRPHYC_PTR0)

Address offset: 0x018

Reset value: 0x0022 AF9B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TITMSRST[3:0]				TDLLLOCK [11:10]	
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDLLLOCK[9:0]										TDLLSRST[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:18 **TITMSRST[3:0]**: ITM soft reset

Number of configuration clock cycles that the ITM soft reset pin must remain asserted when the soft reset is applied to the ITMs.

This must correspond to a value that is equal to or more than 8 controller clock cycles.

Default value corresponds to 8 controller clock cycles.

Bits 17:6 **TDLLLOCK[11:0]**: DLL lock time

Number of configuration clock cycles for the DLL to stabilize and lock, i.e. number of clock cycles from when the DLL reset pin is de-asserted to when the DLL has locked and is ready for use.

Default value corresponds to 5.12us at 533MHz.

Bits 5:0 **TDLLSRST[5:0]**: DLL soft reset

Number of configuration clock cycles that the DLL soft reset pin must remain asserted when the soft reset is triggered through the PHY initialization register (DDRPHYC_PIR).

This must correspond to a value that is equal to or more than 50ns or 8 controller clock cycles, whichever is bigger.

Default value corresponds to 50 ns at 533 MHz.

7.7.8 DDRPHYC PT register 1 (DDRPHYC_PTR1)

Address offset: 0x01C

Reset value: 0x0604 111D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TDINIT1[7:0]								TDINIT0[18:16]		
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDINIT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:19 **TDINIT1[7:0]**: tDINIT1
DRAM initialization time corresponding to the following:
DDR3 = CKE high time to first command (tRFC +10 ns or 5 tCK, whichever value is larger)
LPDDR2 = CKE low time with power and clock stable (100 ns)
LPDDR3 = CKE low time with power and clock stable (100 ns)
Default value corresponds to DDR3 360ns at 533MHz.

Bits 18:0 **TDINIT0[18:0]**: tDINIT0
DRAM initialization time corresponding to the following:
DDR3 = CKE low time with power and clock stable (500 us)
LPDDR2 = CKE high time to first command (200 us)
LPDDR3 = CKE high time to first command (200us)
Default value corresponds to DDR3 500 us at 533MHz.

7.7.9 DDRPHYC PT register 2 (DDRPHYC_PTR2)

Address offset: 0x020

Reset value: 0x042D A072

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TDINIT3[9:0]										TDINIT 2[16]
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDINIT2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:17 **TDINIT3[9:0]**: tDINIT3
DRAM initialization time corresponding to the following:
LPDDR2 = Time from ZQ initialization command to first command (1 us)
LPDDR3 = Time from ZQ initialization command to first command (1us)
Default value corresponds to the LPDDR2/3 1 us at 533MHz.

Bits 16:0 **TDINIT2[16:0]**: tDINIT2
DRAM initialization time corresponding to the following:
DDR3 = Reset low time (200 us on power-up or 100 ns after power-up)
LPDDR2 = Time from reset command to end of auto initialization (1 us + 10 us = 11us)
LPDDR3 = Time from reset command to end of auto initialization (11us)
Default value corresponds to DDR3 200 us at 533MHz

7.7.10 DDRPHYC ACIOC register (DDRPHYC_ACIOCR)

Address offset: 0x024

Reset value: 0x3040 0812

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACSR[1:0]		RSTIOM	RSTPDR	RSTPDD	RSTODT	Res.	Res.	Res.	RANKPDR	Res.	Res.	Res.	CSPDD	Res.	Res.
rw	rw	rw	rw	rw	rw				rw				rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	RANKODT	CKPDR[2:0]			CKPDD[2:0]			CKODT[2:0]			ACPDR	ACPDD	ACODT	ACOE	ACIOM
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **ACSR[1:0]**: AC slew rate
Selects slew rate of the I/O for all address and command pins.

Bit 29 **RSTIOM**: Reset I/O mode
Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for SDRAM Reset.

Bit 28 **RSTPDR**: RST pin power down receiver
Powers down, when set, the input receiver on the I/O for SDRAM RST# pin.

Bit 27 **RSTPDD**: RST pin power down driver
Powers down, when set, the output driver on the I/O for SDRAM RST# pin.

Bit 26 **RSTODT**: RST pin ODT
Enables, when set, the on-die termination on the I/O for SDRAM RST# pin.

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 **RANKPDR**: Rank power down receiver
Powers down, when set, the input receiver on the I/O CKE, ODT, and CS# pins.
Only RANKPDR[0] is used for single rank

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **CSPDD**: CS power down driver
Powers down, when set, the output driver on the I/O for CS# pins.
Only PDD[0] is used for single rank.
CKE and ODT driver power down is controlled by DSGCR register.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **RANKODT**: Rank ODT
Enables, when set, the on-die termination on the I/O for CKE, ODT, and CS# pins.

Bits 13:11 **CKPDR[2:0]**: CK pin power down receiver
Powers down, when set, the input receiver on the I/O for CK[0], CK[1], and CK[2] pins, respectively

Bits 10:8 **CKPDD[2:0]**: CK pin power down driver
Powers down, when set, the output driver on the I/O for CK[0], CK[1], and CK[2] pins, respectively.

Bits 7:5 **CKODT[2:0]**: CK pin ODT
Enables, when set, the on-die termination on the I/O for CK[0], CK[1], and CK[2] pins, respectively.

Bit 4 **ACPDR**: AC pins power down receivers
Powers down, when set, the input receiver on the I/O for RAS#, CAS#, WE#, BA[2:0], and A[15:0] pins.

Bit 3 **ACPDD**: AC pins power down drivers
Powers down, when set, the output driver on the I/O for RAS#, CAS#, WE#, BA[2:0], and A[15:0] pins

Bit 2 **ACODT**: AC pins ODT
Enables, when set, the on-die termination on the I/O for RAS#, CAS#, WE#, BA[2:0], and A[15:0] pins

Bit 1 **ACOE**: AC pins output enable
Enables, when set, the output driver on the I/O for all address and command pins

Bit 0 **ACIOM**: AC pins I/O mode
Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for all address and command pins,

7.7.11 DDRPHYC DXCC register (DDRPHYC_DXCCR)

Address offset: 0x028

Reset value: 0x0000 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDT
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RVSEL	DQSNRST	Res.	Res.	DQSNRES[3:0]				DQSRES[3:0]				DXPDR	DXPDD	DXIOM	DXODT
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AWDT**: Active window data train
Indicates if set that data training (DQS gate training and read valid training) should be performed with active DQS gate window.
This is just for debug purposes.
The default is to perform training with passive windowing.

Bit 15 **RVSEL**: ITMD read valid select
Selects the scheme used for ITMD read valid. Valid values are:
0: ITMD read valid signal is generated by delayed DFI read enable signal.
1: ITMD read valid is generated by the ITMD itself using asynchronous crossing.

Bit 14 **DQSNRST**: DQS reset
Indicates, if set, that the ITMS of DQS# should always be put in reset such that its output enable is always '1' and its data output is always '0'.
This is done by driving the oe_set_b and do_rst_b pins of this ITMS to '0' in order to force the unused DQS# PAD to a known state of '0' in applications that don't use DQS#.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **DQSNRES[3:0]**: DQS# resistor

Selects the on-die pull-up/pull-down resistor for DQS# pins. Same encoding as DQSRES.

Note: DQS# resistor must be connected for LPDDR2.

Bits 7:4 **DQSRES[3:0]**: DQS resistor

Selects the on-die pull-down/pull-up resistor for DQS pins.
DQSRES[3] selects pull-down (when set to 0) or pull-up (when set to 1).
DQSRES[2:0] selects the resistor value as follows:
000: Open: On-die resistor disconnected
001: 688 Ω
010: 611 Ω
011: 550 Ω
100: 500 Ω
101: 458 Ω
110: 393 Ω
111: 344 Ω

Note: DQS resistor must be connected for LPDDR2

Bit 3 **DXPDR**: Data power down receiver

Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. It also powers down the PDQSR cell of all DAXT8 macros. This bit is ORed with the PDR and DQSRPD configuration bits of the individual DATX8

Bit 2 **DXPDD**: Data power down driver

Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the PDD configuration bit of the individual DATX8.

Bit 1 **DXIOM**: Data I/O mode

Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) of the I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the IOM configuration bit of the individual DATX8.

Bit 0 **DXODT**: Data on die termination

Enables, when set, the on-die termination on the I/O for DQ, DM, and DQS/DQS# pins of all DATX8 macros. This bit is ORed with the ODT configuration bit of the individual DATX8.

7.7.12 DDRPHYC DSGC register (DDRPHYC_DSGCR)

Address offset: 0x02C

Reset value: 0xFA00 001F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKEOE	RSTOE	ODTOE	CKOE	TPDOE	TPDPD	NL2OE	NL2PD	Res.	Res.	Res.	ODTPDD	Res.	Res.	Res.	CKEPDD
rw	rw	rw	rw	rw	rw	rw	rw				rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FXDLAT	NOBUB	DQSGE[2:0]			DQSGX[2:0]			LPDLLPD	LPIOPD	ZUEN	BDISEN	PUREN
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CKEOE**: CKE output enable
Sets the output driver on the I/O for SDRAM CKE pins.

Bit 30 **RSTOE**: RST output enable
Enables, when set, the output driver on the I/O for SDRAM RST# pin

Bit 29 **ODTOE**: ODT output enable
Enables, when set, the output driver on the I/O for SDRAM ODT pins.

Bit 28 **CKOE**: CK output enable
Enables, when set, the output driver on the I/O for SDRAM CK/CK# pins.

Bit 27 **TPDOE**: TPD output enable (N/A LPDDR only)

Bit 26 **TPDPD**: TPD power down driver (N/A LPDDR only)

Bit 25 **NL2OE**: Non LPDDR2 output enable
Enables, when set, the output driver on the I/O for non-LPDDR2/LPDDR3 (ODT, RAS#, CAS#, WE#, and BA) pins.
This may be used when a chip that is designed for both LPDDR2/LPDDR3 and other DDR modes is being used in LPDDR2/LPDDR3 mode.
For these pins, the I/O output enable signal (OE) is an AND of this bit and the respective output enable bit in ACIOCR or DSGCR registers.

Bit 24 **NL2PD**: Non LPDDR2 power down
Powers down, when set, the output driver and the input receiver on the I/O for non-LPDDR2/LPDDR3 (ODT, RAS#, CAS#, WE#, and BA) pins.
This may be used when a chip that is designed for both LPDDR2/LPDDR3 and other DDR modes is being used in LPDDR2/LPDDR3 mode.
For these pins, the I/O power down signal (PDD or PDR) is an OR of this bit and the respective power-down bit in ACIOCR register.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ODTPDD**: ODT power down driver
Powers down, when set, the output driver on the I/O for ODT pin.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **CKEPDD**: CKE power down driver
Powers down, when set, the output driver on the I/O for CKE pin.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **FXDLAT**: Fixed latency
Specified whether all reads should be returned to the controller with a fixed read latency. Enabling fixed read latency increases the read latency. Valid values are:
0: Disable fixed read latency
1: Enable fixed read latency

Bit 11 **NOBUB**: No bubble
Specified whether reads should be returned to the controller with no bubbles. Enabling no-bubble reads increases the read latency. Valid values are:
0: Bubbles are allowed during reads
1: Bubbles are not allowed during reads

Bits 10:8 **DQSGE[2:0]**: DQS gate early

Specifies the number of clock cycles for which the DQS gating must be enable dearlier than its normal position. Only applicable when using PDQSR I/O cell, passive DQS gating and no drift compensation.

This field is recommended to be set to zero for all DDR types other than LPDDR2/LPDDR3.

For LPDDR2/LPDDR3 it should be set to (tDQSCKmax - tDQSCKmin) divide by clock period and rounded up. tDQSCKmax and tDQSCKmin can be found in the LPDDR/LPDDR2 vendor datasheet.

Bits 7:5 **DQSGX[2:0]**: DQS gate extension

Specifies the number of clock cycles for which the DQS gating must be extended beyond the normal burst length width. Only applicable when using PDQSR I/O cell, passive DQS gating and no drift compensation.

This field is recommended to be set to zero for all DDR types other than LPDDR2/LPDDR3.

For LPDDR2/LPDDR3 it should be set to (tDQSCKmax- tDQSCKmin) divide by clock period and rounded up. tDQSCKmax and tDQSCKmin can be found in the LPDDR2 vendor datasheet.

Bit 4 **LPDLLPD**: Low power DLL power down

Specifies if set that the PHY should respond to the DFI low power opportunity request and power down the DLL of the PHY if the wakeup time request satisfies the DLL lock time.

For more information on requirements when enabling and disabling the DLL, refer to “bypass mode register control”

Bit 3 **LPIOPD**: Low power I/O power down

Specifies if set that the PHY should respond to the DFI low power opportunity request and power down the I/Os of the PHY.

Bit 2 **ZUEN**: zcal on DFI update request

Specifies if set that the PHY should perform impedance calibration (update) whenever there is a controller initiated DFI update request. Otherwise the PHY ignores an update request from the controller.

Bit 1 **BDISEN**: Byte disable enable

Specifies if set that the PHY should respond to DFI byte disable request. Otherwise the byte disable from the DFI is ignored in which case bytes can only be disabled using the DXnGCR register

Bit 0 **PUREN**: PHY update request enable

Specifies if set, that the PHY should issue PHY initiated DFI update request when there is DQS drift of more than ¾ of a clock cycle within one continuous (back-to-back) read burst.

By default the PHY issues PHY initiated update requests and the controller should respond otherwise the PHY may return erroneous values.

The option to disable it is provided only for silicon evaluation and testing.

7.7.13 DDRPHYC DC register (DDRPHYC_DCR)

Address offset: 0x030

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TPD	RDIMM	UDIMM	DDR2T	NOSRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DDRTYPE[1:0]		MPRDQ	PDQ[2:0]			DDR8BNK	DDRMD[2:0]		
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TPD**: Test power down (N/A LPDDR only)

If set it places the DRAM in deep power down mode.

Bit 30 **RDIMM**: Registered DIMM

Indicates if set that a registered DIMM is used. In this case, the PUBL increases the SDRAM write and read latencies (WL/RL) by 1.

This only applies to PUBL internal SDRAM transactions. Transactions generated by the controller must make its own adjustments to WL/RL when using a registered DIMM

Bit 29 **UDIMM**: Unbuffered DIMM

Indicates if set that there is address mirroring on the second rank of an un-buffered DIMM (the rank connected to CS#[1]).

In this case, the PUBL re-scrambles the bank and address when sending mode register commands to the second rank.

This only applies to PUBL internal SDRAM transactions. Transactions generated by the controller must make its own adjustments when using an un-buffered DIMM. DCR[NOSRA] must be set if address mirroring is enabled.

Bit 28 **DDR2T**: 2T timing

Indicates if set that 2T timing should be used by PUB internally generated SDRAM transactions.

Bit 27 **NOSRA**: No simultaneous rank access

Specifies if set that simultaneous rank access on the same clock cycle is not allowed. This means that multiple chip select signals should not be asserted at the same time. This may be required on some DIMM systems.

Bits 26:10 Reserved, must be kept at reset value.

Bits 9:8 **DDRTYPE[1:0]**: DDR type (LPDDR2 S4)

Selects the DDR type for the specified LPDDR mode.

Valid values for LPDDR2 are:

00: LPDDR2-S4

01: LPDDR2-S2

Others: Reserved

Bit 7 **MPRDQ**: MPR DQ

Specifies the value that is driven on non-primary DQ pins during MPR reads.
Valid values are:
0: Primary DQ drives out the data from MPR (0-1-0-1); non-primary DQs drive ‘0’
1: Primary DQ and non-primary DQs all drive the same data from MPR (0-1-0-1)

Bits 6:4 **PDQ[2:0]**: Primary DQ(DDR3 Only)

Specifies the DQ pin in a byte that is designated as a primary pin for Multi-Purpose Register (MPR) reads. Valid values are 0 to 7 for DQ[0] to DQ[7], respectively

Bit 3 **DDR8BNK**: DDR 8 banks

Indicates if set that the SDRAM used has 8 banks. tRPA = tRP+1 and tFAW are used for 8-bank DRAMs, other tRPA = tRP and no tFAW is used.

Note: A setting of 1 for DRAMs that have fewer than 8 banks still results in correct functionality but less tighter DRAM command spacing for the parameters described here.

Bits 2:0 **DDRMD[2:0]**: SDRAM DDR mode

011: DDR3
100: LPDDR2 (Mobile DDR2)
101: LPDDR3 (Mobile DDR3)
Others: Reserved

7.7.14 DDRPHYC DTP register 0 (DDRPHYC_DTPR0)

Address offset: 0x034

Reset value: 0x3092 666E

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCCD	TRC[5:0]						TRRD[3:0]				TRAS[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRCD[3:0]				TRP[3:0]				TWTR[2:0]			TRTP[2:0]			TMRD[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TCCD**: tCCDRead to read and write to write command delay

0: BL/2 for DDR2 and 4 for DDR3
1: BL/2 + 1 for DDR2 and 5 for DDR3

Bits 30:25 **TRC[5:0]**: tRC

Activate to activate command delay (same bank). Valid values are 2 to 42.

Bits 24:21 **TRRD[3:0]**: tRRD

Activate to activate command delay (different banks). Valid values are 1 to 8.

Bits 20:16 **TRAS[4:0]**: tRAS

Activate to precharge command delay. Valid values are 2 to 31.

Bits 15:12 **TRCD[3:0]**: tRCD

Activate to read or write delay. Minimum time from when an activate command is issued to when a read or write to the activated row can be issued. Valid values are 2 to 11.

Bits 11:8 **TRP[3:0]**: tRP
Precharge command period: The minimum time between a precharge command and any other command.
Note: Note that the controller automatically derives tRPA for 8-bank DDR2 devices by adding 1 to tRP. Valid values are 2 to 11.

Bits 7:5 **TWTR[2:0]**: tWTR
Internal write to read command delay. Valid values are 1 to 6.

Bits 4:2 **TRTP[2:0]**: tRTP
Internal read to precharge command delay. Valid values are 2 to 6. Note that even though RTP does not apply to JEDEC DDR devices, this parameter must still be set to a minimum value of 2 for DDR because the Controller always uses the DDR2 equation, AL + BL/2 + max(RTP,2) – 2, to compute the read to precharge timing (which is BL/2 for JEDEC DDR).

Bits 1:0 **TMRD[1:0]**: tMRD
Load mode cycle time: The minimum time between a load mode register command and any other command.
For DDR3 this is the minimum time between two load mode register commands.
For DDR3, the value used for tMRD is 4 plus the value programmed in these bits, i.e. tMRD value for DDR3 ranges from 4 to 7.
For LPDDR3, the value used for tMRD is 8 plus the value programmed in these bits.

7.7.15 DDRPHYC DTP register 1 (DDRPHYC_DTPR1)

Address offset: 0x038

Reset value: 0x0983 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TDQSCKMAX[2:0]			TDQSCKMIN[2:0]			TRFC[7:0]							
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TRTODT	TMOD[1:0]		TFAW[5:0]						TRTW	TAOND[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:27 **TDQSCKMAX[2:0]**: tDQSCKmax
Maximum DQS output access time from CK/CK# (LPDDR2 only).
This value is used for implementing read-to-write spacing. Valid values are 1 to 7.

Bits 26:24 **TDQSCKMIN[2:0]**: tDQSCKmin
DQS output access time from CK/CK# (LPDDR2/3 only).
This value is used for computing the read latency. Valid values are 0 to 7.
This value is derived from the corresponding parameter in the SDRAM datasheet divided by the clock cycle time without rounding up.
The fractional remainder is automatically adjusted for by data training in quarter clock cycle units.
If data training is not performed then this fractional remainder must be converted to quarter clock cycle units and the gating registers (DXnDQSTR) adjusted accordingly.

Bits 23:16 **TRFC[7:0]**: tRFC

Refresh-to-Refresh: Indicates the minimum time, in clock cycles, between two refresh commands or between a refresh and an active command. This is derived from the minimum refresh interval from the datasheet, tRFC(min), divided by the clock cycle time.

The default number of clock cycles is for the largest JEDEC tRFC(min parameter value supported.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRTODT**: tRTODT

Read to ODT delay (DDR3 only). Specifies whether ODT can be enabled immediately after the read post-amble or one clock delay has to be added.

Valid values are:

0: ODT may be turned on immediately after read post-amble

1: ODT may not be turned on until one clock after the read post-amble

If tRTODT is set to 1, then the read-to-write latency is increased by 1 if ODT is enabled.

Bits 10:9 **TMOD[1:0]**: tMOD

Load mode update delay (DDR3 only). The minimum time between a load mode register command and a non-load mode register command.

Valid values are:

00: 12

01: 13

10: 14

11: 15

Bits 8:3 **TFAW[5:0]**: tFAW

4-bank activate period. No more than 4-bank activate commands may be issued in a given tFAW period. Only applies to 8-bank devices.

Valid values are 2 to 31.

Bit 2 **TRTW**: tRTW

Read to Write command delay.

Valid values are:

0: standard bus turn around delay

1: add 1 clock to standard bus turn around delay

This parameter allows the user to increase the delay between issuing Write commands to the SDRAM when preceded by Read commands. This provides an option to increase bus turn-around margin for high frequency systems.

Bits 1:0 **TAOND[1:0]**: tAOND/tAOFD

ODT turn-on/turn-off delays (DDR2 only). The delays are in clock cycles.

Valid values are:

00: 2/2.5

01: 3/3.5

10: 4/4.5

11: 5/5.5

Most DDR2 devices utilize a fixed value of 2/2.5. For non-standard SDRAMs, the user must ensure that the operational Write Latency is always greater than or equal to the ODT turn-on delay. For example, a DDR2 SDRAM with CAS latency set to 3 and CAS additive latency set to 0 has a Write Latency of 2. Thus 2/2.5 can be used, but not 3/3.5 or higher.

7.7.16 DDRPHYC DTP register 2 (DDRPHYC_DTPR2)

Address offset: 0x03C

Reset value: 0x1001 A0C8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TDLLK[9:0]										TCKE[3:1]		
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCKE [0]	TXP[4:0]					TXS[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:19 **TDLLK[9:0]**: tDLLK

DLL locking time. Valid values are 2 to 1023.

Bits 18:15 **TCKE[3:0]**: tCKE

CKE minimum pulse width. Also specifies the minimum time that the SDRAM must remain in power down or self refresh mode. For DDR3 this parameter must be set to the value of tCKESR which is usually bigger than the value of tCKE. Valid values are 2 to 15.

Bits 14:10 **TXP[4:0]**: tXP

Power down exit delay. The minimum time between a power down exit command and any other command. This parameter must be set to the maximum of the various minimum power down exit delay parameters specified in the SDRAM datasheet, i.e. max(tXP, tXARD, tXARDS) for DDR2 and max(tXP, tXPDLL) for DDR3. Valid values are 2 to 31.

Bits 9:0 **TXS[9:0]**: tXS

Self refresh exit delay. The minimum time between a self refresh exit command and any other command. This parameter must be set to the maximum of the various minimum self refresh exit delay parameters specified in the SDRAM datasheet, i.e. max(tXSNR, tXSRD) for DDR2 and max(tXS, tXSDLL) for DDR3. Valid values are 2 to 1023.

7.7.17 DDRPHYC MR0 register for DDR3 (DDRPHYC_DDR3_MR0)

Address offset: 0x040

Reset value: 0x0A52

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD[2:0]			PD	WR[2:0]			DR	TM	CL[3:1]			BT	CL0	BL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 **RSVD[2:0]**: JEDEC reserved.

Bit 12 **PD**: Power-down control

Controls the exit time for power-down modes. Refer to SDRAM datasheet for details on power-down modes. Valid values are:
0: Slow exit (DLL off)
1: Fast exit (DLL on)

Bits 11:9 **WR[2:0]**: Write recovery

This is the value of the write recovery in clock cycles. It is calculated by dividing the data sheet write recovery time, tWR (ns) by the data sheet clock cycle time, tCK (ns) and rounding up a non-integer value to the next integer.
Valid values are:
001: 5
010: 6
011: 7
100: 8
101: 10
110: 12
All other settings are reserved and should not be used.

Note: tWR (ns) is the time from the first SDRAM positive clock edge after the last data-in pair of a write command, to when a precharge of the same bank can be issued.

Bit 8 **DR**: DLL reset (autoclear)

Writing a ‘1’ to this bit resets the SDRAM DLL. This bit is self clearing, i.e. it returns back to ‘0’ after the DLL reset has been issued.

Bit 7 **TM**: Operating mode

Selects either normal operating mode (0) or test mode (1). Test mode is reserved for the manufacturer and should not be used.

Bit 3 **BT**: Burst type

Indicates whether a burst is sequential (0) or interleaved (1).

Bits 6, 5, 4, 2 **CL[3:0]:** CAS latency

The delay, in clock cycles, between when the SDRAM registers a read command to when data is available.
Valid values for CL[3:0] are:
0010: 5
0100: 6
0110: 7
1000: 8
1010: 9
1100: 10
1110: 11
Others: Reserved

Bits 1:0 **BL[1:0]**: Burst length

Determines the maximum number of column locations that can be accessed during a given read or write command.
Valid values for DDR3 are:
00: 8 (Fixed)
01: 4 or 8 (On the fly)
10: 4 (Fixed)
11: Reserved

7.7.18 DDRPHYC MR1 register for DDR3 (DDRPHYC_DDR3_MR1)

Address offset: 0x044

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	QOFF	TDQS	Res.	RTT2	Res.	LEVEL	RTT1	DIC1	AL[1:0]		RTT0	DIC0	DE
			rw	rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **QOFF**: Output enable/disable

0: all outputs function as normal

1: all SDRAM outputs are disabled removing output buffer current.

This feature is intended to be used for IDD characterization of read current and should not be used in normal operation.

Bit 11 **TDQS**: Termination data strobe

When enabled (‘1’) TDQS provides additional termination resistance outputs that may be useful in some system configurations. (N/A)

Bit 10 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **LEVEL**: Write leveling enable (N/A)

Bits 4:3 **AL[1:0]**: Posted CAS Additive Latency:

Setting additive latency that allows read and write commands to be issued to the SDRAM earlier than normal (refer to SDRAM datasheet for details). Valid values are:

00: 0 (AL disabled)

01: CL - 1

10: CL - 2

11: Reserved

Bits 9, 6, 2 **RTT[2:0]**: On die termination
Selects the effective resistance for SDRAM on die termination.
Valid values for {RTT2,RTT1,RTT0} are:
000: ODT disabled
001: RZQ/4
010: RZQ/2
011: RZQ/6
100: RZQ/12
101: RZQ/8
Others: Reserved

Bits 5, 1 **DIC[1:0]**: Output driver impedance control
Controls the output drive strength.
Valid values for {DIC1,DIC0} are:
00: RZQ/6
01: RZQ/7
10: Reserved
11: Reserved

Bit 0 **DE**: DLL enable/disable
Enable (0) or disable (1) the DLL.
DLL must be enabled for normal operation.

7.7.19 DDRPHYC MR1 register for LPDDR2 [alternate] (DDRPHYC_LPDDR2_MR1)

Address offset: 0x044

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWR[2:0]			WC	BT	BL[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:5 **NWR[2:0]**: Write recovery
001: nWR=3
100: nWR=6
110: nWR=8
111: nWR=9
Others: Reserved

Bit 4 **WC**: Wrap control
0: wrap
1: no wrap

Bit 3 **BT**: Burst Type: Indicates whether a burst is sequential (0) or interleaved (1). ,

Bits 2:0 **BL[2:0]**: Burst length
Determines the maximum number of column locations that can be accessed during a given read or write command. Valid values are:
010: 4
011: 8
100: 16
Others: Reserved

7.7.20 DDRPHYC MR1 register for LPDDR3 [alternate] (DDRPHYC_LPDDR3_MR1)

Address offset: 0x044

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWR[2:0]			Res.	Res.	BL[2:0]		
								rw	rw	rw			rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:5 **NWR[2:0]**: Write recovery
If nWRE (MR2 [4]) = 0:
- 001: nWR=3
- 100: nWR=6
- 110: nWR=8
- 111: nWR=9
If nWRE (MR2[4]) = 1:
- 000: nWR=10
- 001: nWR=11
- 010: nWR=12
Others: Reserved

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:0 **BL[2:0]**: Burst length
Determines the maximum number of column locations that can be accessed during a given read or write command.
011: 8
Others: Reserved

7.7.21 DDRPHYC MR2 register for DDR3 (DDRPHYC_DDR3_MR2)

Address offset: 0x048

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RTTWR[1:0]		Res.	SRT	ASR	CWL[2:0]			PASR[2:0]		
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:9 **RTTWR[1:0]**: Dynamic ODT

Selects RTT for dynamic ODT.
00: Dynamic ODT off
01: RZQ/4
10: RZQ/2
11: Reserved

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SRT**: Self-refresh temperature range

Selects either normal (‘0’) or extended (‘1’) operating temperature range during self-refresh.

Bit 6 **ASR**: Auto self-refresh

When enabled (‘1’), SDRAM automatically provides self-refresh power management functions for all supported operating temperature values.
Otherwise the SRT bit must be programmed to indicate the temperature range.

Bits 5:3 **CWL[2:0]**: CAS write latency

The delay, in clock cycles, between when the SDRAM registers a write command to when write data is available.
000: 5 (tCK = 2.5ns)
001: 6 (2.5ns > tCK = 1.875ns)
010: 7 (1.875ns > tCK = 1.5ns)
011: 8 (1.5ns > tCK = 1.25ns)
Others: Reserved

Bits 2:0 **PASR[2:0]**: Partial array self-refresh

Specifies that data located in areas of the array beyond the specified location are lost if self refresh is entered.
Valid settings for 8 banks are:
000: Full Array
001: Half Array (BA[2:0] = 000, 001, 010 & 011)
010: Quarter Array (BA[2:0] = 000, 001)
011: 1/8 Array (BA[2:0] = 000)
100: 3/4 Array (BA[2:0] = 010, 011, 100, 101, 110 & 111)
101: Half Array (BA[2:0] = 100, 101, 110 & 111)
110: Quarter Array (BA[2:0] = 110 & 111)
111: 1/8 Array (BA[2:0] 111)

7.7.22 DDRPHYC MR2 register for LPDDR2 [alternate] (DDRPHYC_LPDDR2_MR2)

Address offset: 0x048

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLWL[2:0]		
													rw	rw	rw

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:0 **RLWL[2:0]**: Read and write latency
001: RL = 3 /WL = 1
010: RL = 4 / WL = 2
011: RL = 5 / WL = 2
100: RL = 6 / WL = 3
101: RL = 7 / WL = 4
110: RL = 8 / WL = 4
Others: Reserved

7.7.23 DDRPHYC MR2 register for LPDDR3 [alternate] (DDRPHYC_LPDDR3_MR2)

Address offset: 0x048

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR	WL	Res.	NWRE	Res.	RLWL[2:0]		
								rw	rw		rw		rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **WR**: New for LPDDR3 (not used by this PHY, leave at zero)

Bit 6 **WL**: New for LPDDR3 (not used by this PHY, leave at zero)

Bit 5 Reserved, must be kept at reset value.

Bit 4 **NWRE:** New for LPDDR3 (not used by this PHY, leave at zero)

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **RLWL[2:0]**: Read and write latency
001: RL = 3 /WL = 1
010: RL = 4 / WL = 2
011: RL = 5 / WL = 2
100: RL = 6 / WL = 3
101: RL = 7 / WL = 4
110: RL = 8 / WL = 4
Others: Reserved

7.7.24 DDRPHYC MR3 register for DDR3 (DDRPHYC_DDR3_MR3)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPR	MPRLOC[1:0]	
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **MPR**: Multi-purpose register enable
Enables, if set, that read data should come from the Multi-Purpose Register. Otherwise read data come from the DRAM array.

Bits 1:0 **MPRLOC[1:0]**: Multi-purpose register (MPR) location
Selects MPR data location.
00: Predefined pattern for system calibration
Others: Reserved

7.7.25 DDRPHYC MR3 register for LPDDR2 [alternate] (DDRPHYC_LPDDR2_MR3)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **DS[3:0]**: Drive strength
Valid values are:
0000: Reserved
0001: 34.3 Ω typical
0010: 40 Ω typical
0011: 48 Ω typical
0100: 60 Ω typical
0101: Reserved
0110: 80 Ω typical
0111: 120 Ω typical
Others: Reserved

7.7.26 DDRPHYC MR3 register for LPDDR3 [alternate] (DDRPHYC_LPDDR3_MR3)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDCTL[1:0]		DQ[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **PDCTL[1:0]**: Power-down control
- 00: ODT disabled by DRAM during power-down
- 01: ODT enabled by DRAM during power-down
- Others: Reserved

Bits 1:0 **DQ[1:0]**: ODT on-die termination
- 00: Disable (default)
- 01: Rzq/4 (see note)
- 10: Rzq/2
- 11: Rzq/1

Note: Rzq/4 is optional for LPDDR3-1066 devices.

7.7.27 DDRPHYC ODTC register (DDRPHYC_ODTCR)

Address offset: 0x050

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRODT
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDODT
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **WRODT**:
Specifies whether ODT should be enabled (‘1’) or disabled (‘0’) on write

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RDODT**:
Specifies whether ODT should be enabled (‘1’) or disabled (‘0’) on read

7.7.28 DDRPHYC DTA register (DDRPHYC_DTAR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTMPR	DTBANK[2:0]			DTROW[15:4]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DTROW[3:0]				DTCOL[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DTMPR**: Data training using MPR (DDR3 Only):

Specifies, if set, that data-training should use the SDRAM Multi-Purpose Register (MPR) register. Otherwise data-training is performed by first writing to some locations in the SDRAM and then reading the back.

Bits 30:28 **DTBANK[2:0]**: Data training bank address:

Selects the SDRAM bank address to be used during data training.

Bits 27:12 **DTROW[15:0]**: Data training row address:

Selects the SDRAM row address to be used during data training.

Bits 11:0 **DTCOL[11:0]**: Data training column address:

Selects the SDRAM column address to be used during data training.
The lower four bits of this address must always be “0000”.

7.7.29 DDRPHYC DTD register 0 (DDRPHYC_DTDR0)

Address offset: 0x058

Reset value: 0xDD22 EE11

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTBYTE3[7:0]								DTBYTE2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DTBYTE1[7:0]								DTBYTE0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DTBYTE3[7:0]**: Data training data

The first 4 bytes (e.g. shifted as Byte 0,1,2,3) of data used during data training. This same data byte is used for each Byte Lane.
Default sequence is a walking 1 while toggling data every data cycle.

Bits 23:16 **DTBYTE2[7:0]**: Data Training Data

Bits 15:8 **DTBYTE1[7:0]**: Data Training Data

Bits 7:0 **DTBYTE0[7:0]**: Data Training Data

7.7.30 DDRPHYC DTD register 1 (DDRPHYC_DTDR1)

Address offset: 0x05C

Reset value: 0x7788 BB44

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DTBYTE7[7:0]								DTBYTE6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTBYTE5[7:0]								DTBYTE4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DTBYTE7[7:0]**: Data training data:
The second 4 bytes (e.g. shifted as Byte 4,5,6,7) of data used during data training. This same data byte is used for each Byte Lane.
Default sequence is a walking 1 while toggling data every data cycle.

Bits 23:16 **DTBYTE6[7:0]**: Data Training Data

Bits 15:8 **DTBYTE5[7:0]**: Data Training Data

Bits 7:0 **DTBYTE4[7:0]**: Data Training Data

7.7.31 DDRPHYC general purpose register 0 (DDRPHYC_GPR0)

Address offset: 0x178

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **GPR0[31:0]**: General purpose register 0 bits

7.7.32 DDRPHYC general purpose register 1 (DDRPHYC_GPR1)

Address offset: 0x17C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GPR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **GPR1[31:0]**: General purpose register 1 bits

7.7.33 DDRPHYC ZQ0C register 0 (DDRPHYC_ZQ0CR0)

Address offset: 0x180

Reset value: 0x0000 014A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ZQPD	ZCAL	ZCALBYP	ZDEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZDATA[19:16]			
rw	rw	rw	rw									rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ZDATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ZQPD**: ZCAL power down

Powers down, if set, the PZQ cell.

Bit 30 **ZCAL**: ZCAL trigger

Impedance Calibration Trigger: A write of '1' to this bit triggers impedance calibration to be performed by the impedance control logic.

The impedance calibration trigger bit is self-clearing and returns back to '0' when the calibration is complete.

Note: If ZDEN is set, then the ZCAL bit should be set to 0.

Bit 29 **ZCALBYP**: Impedance calibration bypass

Impedance Calibration Bypass: Disables, if set, impedance calibration of this ZQ control block when impedance calibration is triggered globally using the ZCAL bit of DDRPHYC_PIR. Impedance calibration of this ZQ block may be triggered manually using ZCAL.

Bit 28 **ZDEN**: Impedance override enable

Impedance Over-ride Enable: When this bit is set, it allows users to directly drive the impedance control using the data programmed in the ZQDATA field. Otherwise, the control is generated automatically by the impedance control logic.

Note: If ZDEN is set, then the ZCAL bit should be set to 0.

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:0 **ZDATA[19:0]**: Impedance override

Impedance Over-Ride Data: Data used to directly drive the impedance control.

ZDATA field mapping for **D3R I/Os** is as follows:

ZDATA[27:20] is reserved and returns zeros on reads

ZDATA[19:15] is used to select the pull-up on-die termination impedance

ZDATA[14:10] is used to select the pull-down on-die termination impedance

ZDATA[9:5] is used to select the pull-up output impedance

ZDATA[4:0] is used to select the pull-down output impedance

Note: See mPHY ST40c25 Datasheet for ZDATA encoding, default value is: ODT pulp-up/down set to hi-impedance and pull-up/down set to ~44.5 Ω for DDR3, 48.8 Ω for DDR3L, 55.5 Ω for LPDDR2 typical corner 25C

7.7.34 DDRPHYC ZQ0CR1 register (DDRPHYC_ZQ0CR1)

Address offset: 0x184

Reset value: 0x7B

7	6	5	4	3	2	1	0
ZPROG[7:4]				ZPROG[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw

Bits 7:4 **ZPROG[7:4]**: Impedance divide ratio to ext R

Impedance divide ratio: selects the external resistor divide ratio to be used to set the output impedance and the on-die termination as follows:

ZPROG[7:4] = On-die termination divide select (with RZQ = 240 Ω ±1%)

- 0x0: Reserved
- 0x1: 120 Ω
- 0x2: 96 Ω
- 0x3: 80 Ω
- 0x4: 69 Ω
- 0x5: 60 Ω
- 0x6: 52Ω
- 0x7: 46 Ω
- 0x8: 40 Ω
- 0x9: 37 Ω
- 0xA: 34 Ω
- 0xB: 32 Ω
- 0xC: 30 Ω
- 0xD: 28 Ω
- 0xE: 26.5 Ω
- 0xF: 25 Ω

Bits 3:0 **ZPROG[3:0]**: Impedance divide ratio to ext R

Impedance divide ratio: selects the external resistor divide ratio to be used to set the output impedance and the on-die termination as follows:

ZPROG[3:0] = Output impedance divide select (with RZQ = 240 Ω ±1%)

- 0x0 to 0x4: Reserved
- 0x5: 80 Ω
- 0x6: 69 Ω
- 0x7: 60 Ω
- 0x8: 53 Ω
- 0x9: 48 Ω
- 0xA: 44 Ω
- 0xB: 40 Ω
- 0xC: 37 Ω
- 0xD: 34 Ω
- 0xE: 32 Ω
- 0xF: 30 Ω

7.7.35 DDRPHYC ZQ0S register 0 (DDRPHYC_ZQ0SR0)

Address offset: 0x188

Reset value: 0x0000 014A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ZDONE	ZERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZCTRL[19:16]			
r	r											r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ZCTRL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ZDONE**: Impedance calibration done
Indicates that impedance calibration has completed.

Bit 30 **ZERR**: Impedance calibration error
If set, indicates that there was an error during impedance calibration.

Bits 29:20 Reserved, must be kept at reset value.

Bits 19:0 **ZCTRL[19:0]**: Impedance control
Impedance Control: Current value of impedance control.
ZZCTRL field mapping for D3R I/Os is as follows:
ZCTRL[27:20] is reserved and returns zeros on reads
ZCTRL[19:15] is used to select the pull-up on-die termination impedance
ZCTRL[14:10] is used to select the pull-down on-die termination impedance
ZCTRL[9:5] is used to select the pull-up output impedance
ZCTRL[4:0] is used to select the pull-down output impedance

7.7.36 DDRPHYC ZQ0S register 1 (DDRPHYC_ZQ0SR1)

Address offset: 0x18C

Reset value: 0x00

7	6	5	4	3	2	1	0
OPU[1:0]		OPD[1:0]		ZPU[1:0]		ZPD[1:0]	
r	r	r	r	r	r	r	r

Bits 7:6 **OPU[1:0]**: opu calibration status
On-Die termination (ODT) pull-up calibration status.
00: Completed with no errors
01: Overflow error
10: Underflow error
11: Calibration in progress

Bits 5:4 **OPD[1:0]**: opd calibration status
On-Die termination (ODT) pull-down calibration status.
00: Completed with no errors
01: Overflow error
10: Underflow error
11: Calibration in progress

Bits 3:2 **ZPU[1:0]**: zpu calibration status
Output impedance pull-up calibration status. Valid status encodings are:
00: Completed with no errors
01: Overflow error
10: Underflow error
11: Calibration in progress

Bits 1:0 **ZPD[1:0]**: zpd calibration status
Output impedance pull-down calibration status. Valid status encodings are:
00: Completed with no errors
01: Overflow error
10: Underflow error
11: Calibration in progress

7.7.37 DDRPHYC byte lane n GC register (DDRPHYC_DXnGCR)

Address offset: 0x1C0 + 0x40 * n, (n = 0 to 3)

Reset value: 0x0001 0E81

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0RVSL[2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0RVSL[1:0]		RTTOAL	RTTOH[1:0]		DQRTT	DQSRTT	DSEN[1:0]		DQSRPD	DXPDR	DXPDD	DXIOM	DQODT	DQSODT	DXEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **R0RVSL[2:0]**: Read valid system latency in steps

Used to specify the read valid system latency relative to the ideal placement of the ITMD read valid signal when bit RVSEL in DDRPHYC_DXCCR register is set to 0.

Power-up default is 011 (i.e. ideal placement of the read valid signal).

The RVSL fields are initially set by the PUB during automatic read valid training but these values can be overwritten by a direct write to this register.

Valid values are:

000: read valid system latency = ideal placement - 3

001: read valid system latency = ideal placement - 2

010: read valid system latency = ideal placement - 1

011: read valid system latency = ideal placement

100: read valid system latency = ideal placement + 1

101: read valid system latency = ideal placement + 2

110: read valid system latency = ideal placement + 3

Others: Reserved

Bit 13 **RTTOAL**: RTT ON additive latency

Indicates when the ODT control of DQ/DQS SSTL I/Os is set to the value in DQODT/DQSODT during read cycles.

Valid values are:

0: ODT control is set to DQSODT/DQODT almost two cycles before read data preamble

1: ODT control is set to DQSODT/DQODT almost one cycle before read data preamble

Bits 12:11 **RTTOH[1:0]**: RTT output hold

Indicates the number of clock cycles (from 0 to 3) after the read data postamble for which ODT control should remain set to DQSODT for DQS or DQODT for DQ/DM before disabling it (setting it to ‘0’) when using dynamic ODT control.

ODT is disabled almost RTTOH clock cycles after the read postamble

Bit 10 **DQRTT**: DQ dynamic RTT control

Indicates, if set, that the ODT control of DQ/DM SSTL I/Os be dynamically controlled by setting it to the value in DQODT during reads and disabling it (setting it to ‘0’) during any other cycle.

If this bit is not set, then the ODT control of DQ SSTL I/Os is always set to the value in DQODT.

Since dynamic ODT is on by default, when using LPDDR2/LPDDR3 this bit must be set to 0 since LPDDR2/LPDDR3 does not require ODT to be on.`

Bit 9 **DQSRTT**: DQS dynamic RTT control

Indicates, if set, that the ODT control of DQS SSTL I/Os be dynamically controlled by setting it to the value in DQSODT during reads and disabling it (setting it to ‘0’) during any other cycle.

If this bit is not set, then the ODT control of DQS SSTL I/Os is always set to the value in DQSODT field.

Since dynamic ODT is on by default, when using LPDDR2/LPDDR3 this bit must be set to 0 since LPDDR2/LPDDR3 does not require ODT to be on.

Bits 8:7 **DSEN[1:0]**: Write DQS enable
Controls whether the write DQS going to the SDRAM is enabled (toggling) or disabled (static value) and whether the DQS is inverted.
DQS# is always the inversion of DQS.
These values are valid only when DQS/DQS# output enable is on, otherwise the DQS/DQS# is tristated.
Valid settings are:
00: DQS disabled (Driven to constant 0)
01: DQS toggling with normal polarity (This should be the default setting)
10: DQS toggling with inverted polarity
11: DQS disabled (Driven to constant 1)

Bit 6 **DQSRPD**: DQSR power-down
Powers down, if set, the PDQSR cell.
This bit is ORed with the common PDR configuration bit

Bit 5 **DXPDR**: Data power-down receiver
Powers down, when set, the input receiver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDR configuration bit

Bit 4 **DXPDD**: Data power-down driver
Powers down, when set, the output driver on I/O for DQ, DM, and DQS/DQS# pins of the byte. This bit is ORed with the common PDD configuration bit

Bit 3 **DXIOM**: Data I/O mode
Selects SSTL mode (when set to 0) or CMOS mode (when set to 1) for the I/O for DQ, DM, and DQS/DQS# pins of the byte.
This bit is ORed with the IOM configuration bit of the individual DATX8

Bit 2 **DQODT**: DQ ODT enable
Enables, when set, the on-die termination on the I/O for DQ and DM pins of the byte.
This bit is ORed with the common DATX8 ODT configuration bit

Bit 1 **DQSODT**: DQS ODT enable
Enables, when set, the on-die termination on the I/O for DQS/DQS# pin of the byte.
This bit is ORed with the common DATX8 ODT configuration bit

Bit 0 **DXEN**: DATA byte enable
Enables, if set, the DATX8 and SSTL I/Os used on the data byte.
Setting this bit to ‘0’ disables the byte, i.e. the byte SSTL I/Os are put in power-down mode and the DLL in the DATX8 is put in bypass mode.
After changing a Byte Lane from disabled to enabled, the DLL for that Byte Lane must be reset and re-locked.
Software can use bits DLLSRST and DLLLOCK of DDRPHYC_PIR register to accomplish this (reset and re-locks all DLLs in the DDR PHY).

7.7.38 DDRPHYC byte lane n GS register 0 (DDRPHYC_DXnGSR0)

Address offset: 0x1C4 + 0x40 *n, (n = 0 to 3)

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTPASS[2:0]			Res.	Res.	Res.	Res.	DTIERR	Res.	Res.	Res.	DTERR	Res.	Res.	Res.	DTDONE
r	r	r					r				r				r

Bits 15:13 **DTPASS[2:0]**: DQS training pass count
The number of passing configurations during DQS gate training.

Bits 12:9 Reserved, must be kept at reset value.

Bit 8 **DTIERR**: DQS gate training intermittent error
If set, indicates that there was an intermittent error during DQS gate training of the byte, such as a pass was followed by a fail then followed by another pass.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **DTERR**: DQS gate training error
If set, indicates that a valid DQS gating window could not be found during DQS gate training of the byte.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DTDONE**: Data training done
Indicates, if set, that the byte has finished doing data training.

7.7.39 DDRPHYC byte lane n GS register 1 (DDRPHYC_DXnGSR1)

Address offset: 0x1C8 + 0x40 * n, (n = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVPASS[2:0]			Res.	Res.	Res.	RVIERR
									r	r	r				r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	RVERR	Res.	Res.	Res.	Res.	Res.	Res.	DQSDFT[1:0]		Res.	Res.	Res.	DFTERR
			r							r	r				r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **RVPASS[2:0]**: Read valid training pass count
The number of passing configurations during read valid training.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RVIERR**: RV intermittent error for rank

If set, indicates that there was an intermittent error during read valid training of the byte, such as a pass was followed by a fail then followed by another pass.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RVERR**: RV training error

If set, indicates that a valid read valid placement could not be found during read valid training of the byte

Bits 11:6 Reserved, must be kept at reset value.

Bits 5:4 **DQSDFT[1:0]**: DQS drift value

Used to report the drift on the read data strobe of the data byte.
Valid settings are:
00: No drift
01: 90° drift
10: 180° drift
11: 270° drift or more

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DFTERR**: DQS drift error

If set, indicates that the byte read data strobe has drifted by more than or equal to the drift limit set in the PHY General Configuration Register (PGCR).

7.7.40 DDRPHYC byte lane n DLLC register (DDRPHYC_DXnDLLCR)

Address offset: 0x1CC + 0x40 * n, (n = 0 to 3)

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLBMODE	ATESTEN	SDPHASE[3:2]	
rw	rw											rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SDPHASE[1:0]		SSTART[1:0]		MFWDLY[2:0]			MFBDLY[2:0]			SFWDLY[2:0]			SFBDLY[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DLLDIS**: DLL bypass

A disabled DLL is bypassed. Default ('0') is DLL enabled.

Note: DLL bypass must only be changed when DDRPHYC clock (dphy_ker_ck) is inactive.

Bit 30 **DLLSRST**: DLL reset

Soft resets the byte DLL by driving the DLL soft reset pin.

Note: DLL reset must only be changed to 0 when DDRPHYC clock (dphy_ker_ck) is inactive.

Bits 29:20 Reserved, must be kept at reset value.

Bit 19 **SDLBMODE**: Bypass slave DLL during loopback

If this bit is set, the slave DLL is put in loopback mode in which there is no 90 degrees phase shift on read DQS/DQS#. This bit must be set when operating the byte PHYs in loopback mode.

Bit 18 **ATESTEN**: Enable path to pin 'ATO'
Enables the analog test signal to be output on the DLL analog test output (DDR_ATO). The DLL analog test output is tri-stated when this bit is '0'.

Bits 17:14 **SDPHASE[3:0]**: Slave DLL phase
Selects the phase difference between the input clock and the corresponding output clock of the slave DLL. Valid settings:
0000: 90
0001: 72
0010: 54
0011: 36
0100: 108
0101: 90
0110: 72
0111: 54
1000: 126
1001: 108
1010: 90
1011: 72
1100: 144
1101: 126
1110: 108
1111: 90

Bits 13:12 **SSTART[1:0]**: Slave DLL autostart
Used to control how the slave DLL starts up relative to the master DLL locking:
0X: Slave DLL automatically starts up once the master DLL has achieved lock.
10: The automatic startup of the slave DLL is disabled; the phase detector is disabled.
11: The automatic startup of the slave DLL is disabled; the phase detector is enabled.

Bits 11:9 **MFWDLY[2:0]**: Master DLL feed-forward trim
Used to trim the delay in the master DLL feedforward path:
000: minimum delay
...
111: maximum delay

Bits 8:6 **MFBDLY[2:0]**: Master DLL feed-back trim
Used to trim the delay in the master DLL feedback path:
000: minimum delay
...
111: maximum delay

Bits 5:3 **SFWDLY[2:0]**: Slave DLL feed-forward trim
Slave Feed-Forward Delay Trim: Used to trim the delay in the slave DLL feedforward path:
000: minimum delay
...
111: maximum delay

Bits 2:0 **SFBDLY[2:0]**: Slave DLL feed-back trim
Slave Feed-Back Delay Trim: Used to trim the delay in the slave DLL feedback path:
000: minimum delay
...
111: maximum delay

7.7.41 DDRPHYC byte lane n DQT register (DDRPHYC_DXnDQTR)

Address offset: 0x1D0 + 0x40 * n, (n = 0 to 3)

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DQDLY7[3:0]				DQDLY6[3:0]				DQDLY5[3:0]				DQDLY4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DQDLY3[3:0]				DQDLY2[3:0]				DQDLY1[3:0]				DQDLY0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **DQDLY7[3:0]**: DQ delay for bit 7
Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree.
The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b.
Valid settings for each 2-bit control field are:
00: nominal delay
01: nominal delay + 1 step
10: nominal delay + 2 steps
11: nominal delay + 3 steps

Bits 27:24 **DQDLY6[3:0]**: DQ delay for bit 6
idem

Bits 23:20 **DQDLY5[3:0]**: DQ delay for bit 5
Idem

Bits 19:16 **DQDLY4[3:0]**: DQ delay for bit 4
Idem

Bits 15:12 **DQDLY3[3:0]**: DQ delay for bit 3
Idem

Bits 11:8 **DQDLY2[3:0]**: DQ delay for bit 2
Idem

Bits 7:4 **DQDLY1[3:0]**: DQ delay for bit 1
Idem

Bits 3:0 **DQDLY0[3:0]**: DQ delay for bit 0
Idem

7.7.42 DDRPHYC byte lane n DQST register (DDRPHYC_DXnDQSTR)

Address offset: 0x1D4 + 0x40 * n, (n = 0 to 3)

Reset value: 0x3DB0 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DMDLY[3:0]				DQSNDLY[2:0]			DQSDLY[2:0]			Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	R0DGPS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0DGSL[2:0]		
		rw	rw										rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:26 **DMDLY[3:0]**: DM delay
Used to adjust the delay of the data relative to the nominal delay that is matched to the delay of the data strobes through the slave DLL and clock tree.
The lower two bits of the DQDLY for each DQ bit controls the delay for the data clocked by DQS, while the higher two bits control the delay for the data clocked by DQS_b.
Valid settings for each 2-bit control field are:
00: nominal delay
01: nominal delay + 1 step
10: nominal delay + 2 steps
11: nominal delay + 3 steps

Bits 25:23 **DQSNDLY[2:0]**: DQS# delay
Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree.
Valid values are:
000: nominal delay - 3 steps
001: nominal delay - 2 steps
010: nominal delay - 1 step
011: nominal delay
100: nominal delay + 1 step
101: nominal delay + 2 steps
110: nominal delay + 3 steps
111: nominal delay + 4 steps

Note: - After changing this value, an ITM soft reset must be issued: bit ITMSRST=1 and bit INIT=1 in DDRPHYC_PIR register.

Bits 22:20 **DQSDLY[2:0]**: DQS delay

Used to adjust the delay of the data strobes relative to the nominal delay that is matched to the delay of the data bit through the slave DLL and clock tree.

Valid values are:

000: nominal delay - 3 steps
001: nominal delay - 2 steps
010: nominal delay - 1 step
011: nominal delay
100: nominal delay + 1 step
101: nominal delay + 2 steps
110: nominal delay + 3 steps
111: nominal delay + 4 steps

Note: - After changing this value, an ITM soft reset (bit ITMSRST=1 and bit INIT=1 in DDRPHYC_PIR register) must be issued.

Bits 19:14 Reserved, must be kept at reset value.

Bits 13:12 **R0DGPS[1:0]**: Rank 0 DQS gating phase select

Selects the clock used to enable the data strobes during read so that the value of the data strobes before and after the preamble/postamble are filtered out.

R0DGPS is initially set by the PUBL during automatic DQS data training and subsequently updated during data strobe drift compensation. However, these values can be overwritten by a direct write to this register, and the automatic update during DQS drift compensation can be disabled using the PHY General Configuration Register (PGCR).

Valid values are:

00: 180 clock (clk180)
01: 270 clock (clk270)
10: 360 clock (clk0)
11: 450 clock (next clk90)

Bits 11:3 Reserved, must be kept at reset value.

Bits 2:0 **R0DGSL[2:0]**: Rank 0 DQS gating system latency

Used to increase the number of clock cycles needed to expect valid DDR read data by up to five extra clock cycles.

This is used to compensate for board delays and other system delays.

Power-up default is 000 (i.e. no extra clock cycles required).

The SL fields are initially set by the PUBL during automatic DQS data training but these values can be overwritten by a direct write to this register.

Valid values are:

000: No extra clock cycles
001: 1 extra clock cycle
010: 2 extra clock cycles
011: 3 extra clock cycles
100: 4 extra clock cycles
101: 5 extra clock cycles
Others: Reserved

7.7.43 DDRPHYC register map

Table 26. DDRPHYC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DDRPHYC_RIDR**	UDRID[7:0]								PHYMJR [3:0]				PHYMDR [3:0]				PHYMNR [3:0]				PUBMJR [3:0]				PUBMDR [3:0]				PUBMNR [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x004	**DDRPHYC_PIR**	INITBYP	ZCALBYP	LCOKBYP	CLRSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLLBYP	ICPC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVTRN	QSTRN	DRAMINIT	DRAMRST	ITMSRST	ZCAL	DLLLOCK	DLLSRST	INIT
	Reset value	0	0	0	0											0	0								0	0	0	0	0	0	0	0	0
0x008	**DDRPHYC_PGCR**	LBMODE	LBGDQS	LBDQSS	RFSHDT [3:0]				PDDISDX	ZKSEL[1:0]		RANKEN [3:0]				IODDRM[1:0]		IOLB	CKINV	CKDV[1:0]		CKEN [2:0]			DTOSEL [3:0]				DFTLMT[1:0]		DFTCMP	DQSCFG	ITMDMD
	Reset value	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	1	0	0
0x00C	**DDRPHYC_PGSR**	TQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVEIRR	RVERR	DFTERR	DTIERR	DTERR	DTDONE	DIDONE	ZCDDONE	DLDONE	IDONE
	Reset value	0																						0	0	0	0	0	0	0	0	0	0
0x010	**DDRPHYC_DLLGCR**	DLLRSVD2[1:0]		LOCKDET	FDTRMSL[1:0]		SBIAS5_3[2:0]			BPS200	SBIAS2_0[2:0]			MBIAS[7:0]								TESTSW	ATC [1:0]		DTC [2:0]			TESTEN	IPUMP[2:0]			DRES[1:0]	
	Reset value	0	0	0	0	0	0	1	1	0	1	1	1	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**DDRPHYC_ACDLLCR**	DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATESTEN	Res.	Res.	Res.	Res.	Res.	Res.	MFWDLY[2:0]			MFBDLY[2:0]			Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	1												0							0	0	0	0	0	0						
0x018	**DDRPHYC_PTR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TITMSRST[3:0]				TDLLLOCK[11:0]												TDLLSRST[5:0]					
	Reset value											1	0	0	0	1	0	1	0	1	0	1	1	1	1	1	0	0	1	1	0	1	1
0x01C	**DDRPHYC_PTR1**	Res.	Res.	Res.	Res.	Res.	TDINIT1[7:0]								TDINIT0[18:0]																		
	Reset value						1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	1	0	1
0x020	**DDRPHYC_PTR2**	Res.	Res.	Res.	Res.	Res.	TDINIT3[9:0]										TDINIT2[16:0]																
	Reset value						1	0	0	0	0	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	1	1	1	0	0	1	0
0x024	**DDRPHYC_ACIOCR**	ACSR[1:0]		RSTIOM	RSTPDR	RSTPDD	RSTODT	RANKPDR[3:0]				CSPDD [3:0]				Res.	Res.	Res.	RANKODT	CKPDR[2:0]			CKPDD[2:0]			CKODT[2:0]			ACPDR	ACPDD	ACODT	ACOE	ACIOM
	Reset value	0	0	1	1	0	0	0	0	0	1	0	0	0	0				0	0	0	1	0	0	0	0	0	0	1	0	0	1	0

Table 26. DDRPHYC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	**DDRPHYC_DXCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDT	RVSEL	DQSNRST	Res.	Res.	DQSNRES[3:0]				DQSRES[3:0]				DXPDR	DXPDD	DXIOM	DXODT
	Reset value																0	0	0			1	0	0	0	0	0	0	0	0	0	0	0
0x02C	**DDRPHYC_DSGCR**	CKOE	RSTOE	ODTOE	CKOE	TPDOE	TPDPD	NL2OE	NL2PD	ODTPDD [3:0]				CKEPDD [3:0]				Res.	Res.	Res.	FXDLAT	NOBUB	DQSGE[2:0]			DQSGX[2:0]			LPDLLPD	LPIOPD	ZUEN	BDISEN	PUREN
	Reset value	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	1	1	1	1	1
0x030	**DDRPHYC_DCR**	TPD	RDIMM	UDIMM	DDR2T	NOSRA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRTYPE[1:0]		MPRDQ	PDQ[2:0]			DDR8BNK	DDRMD[2:0]		
	Reset value	0	0	0	0	0																		0	0	0	0	0	0	1	0	1	1
0x034	**DDRPHYC_DTPR0**	TCCD	TRC[5:0]						TRRD[3:0]				TRAS[4:0]					TRCD[3:0]				TRP[3:0]				TWTR [2:0]			TRTP [2:0]			TMRD[1:0]	
	Reset value	0	0	1	1	0	0	0	0	1	0	0	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	0
0x038	**DDRPHYC_DTPR1**	Res.	Res.	TDQSCKMAX[2:0]			TDQSCKMIN[2:0]			TRFC[7:0]								Res.	Res.	Res.	Res.	TRTODT	TMOD[1:0]		TFAW[5:0]						TRTW	TAOND[1:0]	
	Reset value			0	0	1	0	0	1	1	0	0	0	0	0	1	1					0	0	0	0	1	0	0	1	0	0	0	0
0x03C	**DDRPHYC_DTPR2**	Res.	Res.	Res.	TDLLK[9:0]										TCKE[3:0]				TXP[4:0]					TXS[9:0]									
	Reset value				1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	1	1	0	0	1	0	0	0
0x040	**DDRPHYC_DDR3_MR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSVD [2:0]			PD	WR[2:0]			DR	TM	CL1 [2:0]			BT	CL0	BL [1:0]	
	Reset value																	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0
0x042 - 0x042	Reserved	Res.																															
0x044	**DDRPHYC_DDR3_MR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QOFF	TDQS	Res.	RTT2	Res.	LEVEL	RTT1	DIC1	AL [1:0]		RTT0	DIC0	DE
	Reset value																				0	0		0		0	0	0	0	0	0	0	0
0x044	**DDRPHYC_LPDDR2_ MR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWR [2:0]			WC	BT	BL[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x044	**DDRPHYC_LPDDR3_ MR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWR [2:0]			Res.	Res.	BL[2:0]		
	Reset value																									0	0	0			0	0	0
0x046 - 0x046	Reserved	Res.																															

Table 26. DDRPHYC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	**DDRPHYC_DDR3_MR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTTWR[1:0]		Res.	SRT	ASR	CWL [2:0]			PASR [2:0]		
	Reset value																						0	0		0	0	0	0	0	0	0	0
0x048	**DDRPHYC_LPDDR2_ MR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLWL [2:0]		
	Reset value																														0	0	0
0x048	**DDRPHYC_LPDDR3_ MR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WR	WL	Res.	NWRE	Res.	RLWL [2:0]		
	Reset value																									0	0		0		0	0	0
0x04A - 0x04B	Reserved	Res.																															
0x04C	**DDRPHYC_DDR3_MR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPR	MPRLOC[1:0]	
	Reset value																														0	0	0
0x04C	**DDRPHYC_LPDDR2_ MR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS[3:0]			
	Reset value																													0	0	0	0
0x04C	**DDRPHYC_LPDDR3_ MR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDCTL[1:0]		DQ[1:0]	
	Reset value																													0	0	0	0
0x04D - 0x04E	Reserved	Res.																															
0x050	**DDRPHYC_ODTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRODT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDODT
	Reset value																1																0
0x052 - 0x050	Reserved	Res.																															
0x054	**DDRPHYC_DTAR**	DTMPR	DTBANK[2:0]			DTROW[15:0]																DTCOL[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	**DDRPHYC_DTDR0**	DTBYTE3[7:0]								DTBYTE2[7:0]								DTBYTE1[7:0]								DTBYTE0[7:0]							
	Reset value	1	1	0	1	1	1	0	1	0	0	1	0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	1	0	0	0	1
0x05C	**DDRPHYC_DTDR1**	DTBYTE7[7:0]								DTBYTE6[7:0]								DTBYTE5[7:0]								DTBYTE4[7:0]							
	Reset value	0	1	1	1	0	1	1	1	1	0	0	0	1	0	0	0	1	0	1	1	1	0	1	1	0	1	0	0	0	1	0	0
0x060 - 0x174	Reserved	Res.																															
0x178	**DDRPHYC_GPR0**	GPR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17C	**DDRPHYC_GPR1**	GPR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 26. DDRPHYC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x180	**DDRPHYC_ZQ0CR0**	ZQPD	ZCAL	ZCALBYP	ZDEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZDATA[19:0]																			
	Reset value	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0
0x184	**DDRPHYC_ZQ0CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZPROG[7:4]				ZPROG[3:0]			
	Reset value																									0	1	1	1	1	0	1	1
0x185 - 0x184	Reserved	Res.																															
0x188	**DDRPHYC_ZQ0SR0**	ZDONE	ZERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ZCTRL[19:0]																			
	Reset value	0	0											0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0
0x18C	**DDRPHYC_ZQ0SR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPU [1:0]		OPD [1:0]		ZPU [1:0]		ZPD [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x18D - 0x1BC	Reserved	Res.																															
0x1C0	**DDRPHYC_DX0GCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0RVSL[2:0]			RTTOAL	RTTOH[1:0]		DQRTT	DQSRTT	DSEN[1:0]		DQSRPD	DXPDR	DXPDD	DXIOM	DQODT	DQSODT	DXEN
	Reset value																1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1
0x1C4	**DDRPHYC_DX0GSR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPASS[2:0]			Res.	Res.	Res.	Res.	DTIERR	Res.	Res.	Res.	DTERR	Res.	Res.	Res.	DTDONE
	Reset value																	0	0	0					0				0				0
0x1C8	**DDRPHYC_DX0GSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVPASS[2:0]			Res.	Res.	Res.	RVIERR	Res.	Res.	Res.	RVERR	Res.	Res.	Res.	Res.	Res.	Res.	DQSDFT[1:0]		Res.	Res.	Res.	DFTERR
	Reset value										0	0	0				0				0							0	0				0
0x1CC	**DDRPHYC_DX0DLLCR**	DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLBMODE	ATESTEN	SDPHASE[3:0]				SSTART[1:0]		MFWDLY[2:0]			MFBDLY[2:0]			SFWDLY[2:0]			SFBDLY[2:0]		
	Reset value	0	1											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D0	**DDRPHYC_DX0DQTR**	DQDLY7 [3:0]				DQDLY6 [3:0]				DQDLY5 [3:0]				DQDLY4 [3:0]				DQDLY3 [3:0]				DQDLY2 [3:0]				DQDLY1 [3:0]				DQDLY0 [3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1D4	**DDRPHYC_DX0DQSTR**	Res.	Res.	DMDLY [3:0]				DQSNDLY[2:0]			DQSDLY[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	R0DGPS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0DGSL[2:0]		
	Reset value			1	1	1	1	0	1	1	0	1	1							1	0										0	0	0
0x1D8 - 0x1FC	Res.																																

Table 26. DDRPHYC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x200	**DDRPHYC_DX1GCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0RVSL[2:0]			RTTOAL	RTTOH[1:0]		DQRTT	DQSRTT	DSEN[1:0]		DQSRPD	DXPDR	DXPDD	DXIOM	DQODT	DQSODT	DXEN
	Reset value																1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1
0x204	**DDRPHYC_DX1GSR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVPASS[2:0]			Res.	Res.	Res.	RVIERR	Res.	Res.	Res.	RVERR	Res.	Res.	Res.	Res.	Res.	Res.	DQSDFT[1:0]		Res.	Res.	Res.	DFTERR
	Reset value										0	0	0				0				0							0	0				0
0x208	**DDRPHYC_DX1GSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVPASS[2:0]			Res.	Res.	Res.	RVIERR	Res.	Res.	Res.	RVERR	Res.	Res.	Res.	Res.	Res.		DQSDFT[1:0]		Res.	Res.	Res.	DFTERR
	Reset value										0	0	0				0				0							0	0				0
0x20C	**DDRPHYC_DX1DLLCR**	DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLBMODE	ATESTEN	SDPHASE[3:0]				SSTART[1:0]		MFWDLY[2:0]			MFBDLY[2:0]			SFWDLY[2:0]			SFBDLY[2:0]		
	Reset value	0	1											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	**DDRPHYC_DX1DQTR**	DQDLY7 [3:0]				DQDLY6 [3:0]				DQDLY5 [3:0]				DQDLY4 [3:0]				DQDLY3 [3:0]				DQDLY2 [3:0]				DQDLY1 [3:0]				DQDLY0 [3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x214	**DDRPHYC_DX1DQSTR**	Res.	Res.	DMDLY [3:0]				DQSNDLY[2:0]			DQSDLY[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	R0DGPS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0DGSL[2:0]		
	Reset value			1	1	1	1	0	1	1	0	1	1							1	0										0	0	0
0x218 - 0x23C	Reserved	Res.																															
0x240	**DDRPHYC_DX2GCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0RVSL[2:0]			RTTOAL	RTTOH[1:0]		DQRTT	DQSRTT	DSEN[1:0]		DQSRPD	DXPDR	DXPDD	DXIOM	DQODT	DQSODT	DXEN
	Reset value																1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1
0x246 - 0x244	Reserved	Res.																															
0x248	**DDRPHYC_DX2GSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVPASS[2:0]			Res.	Res.	Res.	RVIERR	Res.	Res.	Res.	RVERR	Res.	Res.	Res.	Res.	Res.	Res.	DQSDFT[1:0]		Res.	Res.	Res.	DFTERR
	Reset value										0	0	0				0				0							0	0				0

Table 26. DDRPHYC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24C	DDRPHYC_DX2DLLCR	DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLBMODE	ATESTEN	SDPHASE[3:0]				SSTART[1:0]		MFWDLY[2:0]			MFBDLY[2:0]			SFWDLY[2:0]			SFBDLY[2:0]		
	Reset value	0	1											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x250	DDRPHYC_DX2DQTR	DQDLY7 [3:0]				DQDLY6 [3:0]				DQDLY5 [3:0]				DQDLY4 [3:0]				DQDLY3 [3:0]				DQDLY2 [3:0]				DQDLY1 [3:0]				DQDLY0 [3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x254	DDRPHYC_DX2DQSTR	Res.	Res.	DMDLY [3:0]				DQSNDLY[2:0]			DQSDLY[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	R0DGPS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0DGSL[2:0]		
	Reset value			1	1	1	1	0	1	1	0	1	1							1	0										0	0	0
0x258 - 0x27C	Reserved	Res.																															
0x280	DDRPHYC_DX3GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0RVSL[2:0]			RTTOAL	RTTOH[1:0]		DQRTT	DQSRTT	DSEN[1:0]		DQSRPD	DXPDR	DXPDD	DXIOM	DQODT	DQSODT	DXEN
	Reset value																1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1
0x286 - 0x284	Reserved	Res.																															
0x288	DDRPHYC_DX3GSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RVPASS[2:0]			Res.	Res.	Res.	RVIERR	Res.	Res.	Res.	RVERR	Res.	Res.	Res.	Res.	Res.	Res.	DQSDFT[1:0]		Res.	Res.	Res.	DFTERR
	Reset value										0	0	0				0				0							0	0				0
0x28C	DDRPHYC_DX3DLLCR	DLLDIS	DLLSRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLBMODE	ATESTEN	SDPHASE[3:0]				SSTART[1:0]		MFWDLY[2:0]			MFBDLY[2:0]			SFWDLY[2:0]			SFBDLY[2:0]		
	Reset value	0	1											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x290	DDRPHYC_DX3DQTR	DQDLY7 [3:0]				DQDLY6 [3:0]				DQDLY5 [3:0]				DQDLY4 [3:0]				DQDLY3 [3:0]				DQDLY2 [3:0]				DQDLY1 [3:0]				DQDLY0 [3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x294	DDRPHYC_DX3DQSTR	Res.	Res.	DMDLY [3:0]				DQSNDLY[2:0]			DQSDLY[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	R0DGPS[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0DGSL[2:0]		
	Reset value			1	1	1	1	0	1	1	0	1	1							1	0										0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

8 DDR performance monitor (DDRPERFM)

8.1 Introduction

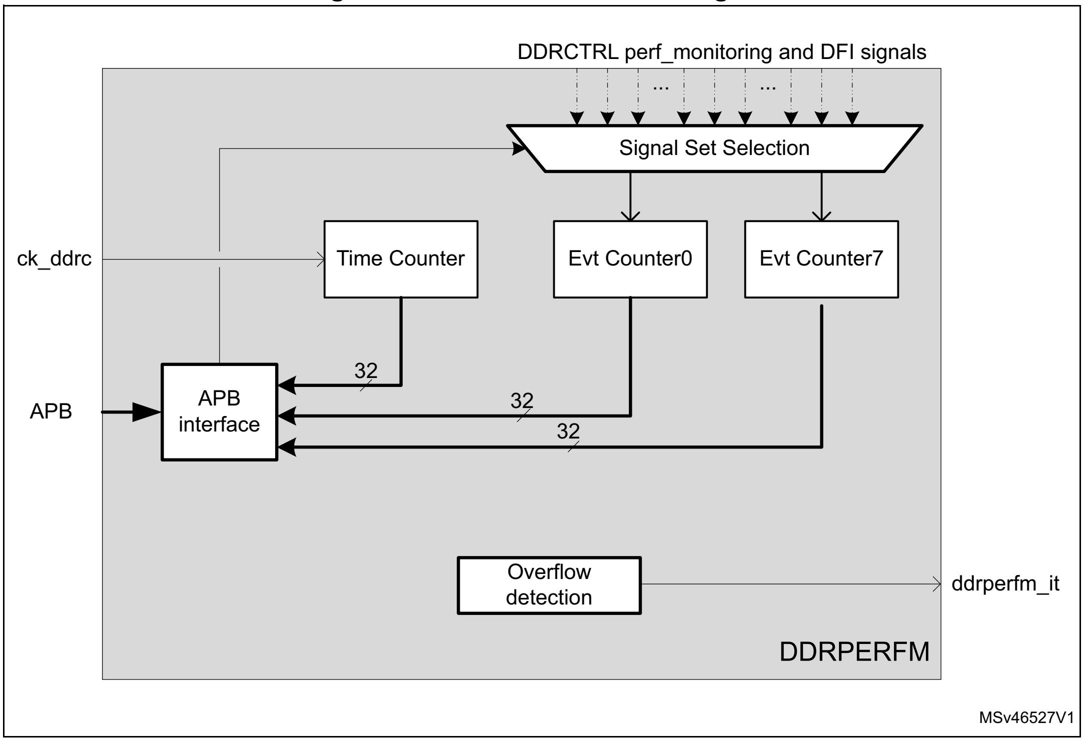
The DDRPERFM is intended to monitor DDR controller performance by counting events from a set of signals and gathering results under software control.
The DDRPERFM can be used to feedback important statistics such as the DDR utilization, read/write bandwidth, bank activation, idle state duration for some off-line analysis of the DDR system performance and for optimal parameter settings to improve the DDR utilization and for power optimization.

8.2 Features

- One 32-bit time counter (TCNT)
- Four 32-bit event counters (CNT0 to CNT3)
- Selection from 4 sets of events to be monitored
- Global Start/ Stop control
- Individual counter enable and clear
- Global counter start and stop triggered by software
- Counter overflow detection with all the counters freeze and the interrupt event generation
- APB interface

8.3 Block diagram

Figure 26. DDRPERFM block diagram



8.4 Functional description

The DDRPERFM provides information of how efficiently the DDR controller is working by reporting statistics according to the set of signals selected for monitoring.

Up to 4 signals from 4 sets of signals along with one permanent clock counter can be monitored.

The counters are started and stopped by software.

All the counters stop when one counter saturates. (The quickest it can saturate is 8.05 seconds with 533 MHz DDR)

The counters may be halted without resetting them to suspend counting.

Useful composite statistics can be derived from counter values; This information can be used by developers to tune the DDR controller settings. For example to optimize the DDR utilization or to improve the low-power mode settings.

Table 27. DDRPERFM signal sets

Set	CNT0	CNT1	CNT2	CNT3
0	perf_op_is_rd	perf_op_is_wr	perf_op_is_activate	ctl_idle
1	perf_hpr_req_with_nocredit	perf_lpr_req_with_nocredit	Reserved	cactive_ddrc
2	perf_op_is_enter_powerdown	perf_op_is_refresh	perf_selfref_mode	dfi_lp_req
3	perf_hpr_xact_when_critical	perf_lpr_xact_when_critical	perf_wr_xact_when_critical	dfi_lp_req

The signals are described hereafter:

perf_op_is_rd	Asserted for every read that is issued for commands going through CAM
perf_op_is_wr	Asserted for every write that is issued for commands going through CAM.
perf_op_is_activate	Asserted for every activate that is issued for commands going through CAM
ctl_idle	Asserted when the controller is idle for extended periods and only when there are no commands or data on the DFI interface.
perf_hpr_req_with_nocredit	Asserted when there is a High Priority Read (HPR) request (from XPI to PA) not served due to no available credit.
perf_lpr_req_with_nocredit	Asserted when there is a Low Priority Read (LPR) request (from XPI to PA) not served due to no available credit.
cactive_ddrc	DDRC Hardware Low-power clock active. Indicates that the DDRC requires its clock signal.
perf_op_is_enter_powerdown	Asserted for every entry into Power down mode
perf_op_is_refresh	Asserted for every refresh command that is issued by the controller.
perf_selfref_mode	Asserted for the entire duration for which the controller stays in Self-Refresh mode.
perf_hpr_xact_when_critical	Asserted for every High Priority Read transaction that is scheduled when the High Priority queue is in critical state.
perf_lpr_xact_when_critical	Asserted for every Low Priority Read transaction that is scheduled when the Low Priority queue is in critical state.
perf_wr_xact_when_critical	Asserted for every write transaction that is scheduled when the write queue is in critical state.
dfi_lp_req	DFI low-power opportunity request. This signal is used by the controller to inform the PHY of an opportunity to switch to a low-power mode.

8.5 Application information

8.5.1 Signal set 0: to monitor the DDR utilization

By counting the read and write commands sent to DDR over a time window, can be monitored:

- The read and write traffic (approximate) bandwidth.
- The DDR utilization.
- The DDR scheduling efficiency

Note: The DDR efficiency is measured by the ratio of read and write count to the time where the DDR is not idle.

Example:

```
read_bandwidth (Mbyte/sec)  ~= 4 * BL * freq * CNT0/TCNT
write_bandwidth (Mbyte/sec) ~= 4 * BL * freq * CNT1/TCNT
DDR_utilization = (BL/2) *(CNT0 + CNT1)/TCNT
with
   BL = 8
   freq = 533 MHz
```

The ratio of activate commands to the total of read and write commands is providing information about the bank conflict efficiency, this information can be used to select the best memory access pattern between the bank sequential or the bank interleaved.

8.5.2 Signal set 1: to monitor the HPR and LPR stores

By counting the durations where the DDRC stores are out-of-credit to fine tune the DDRCTRL scheduling and performance control:

- To better balance the store split between HPR and LPR.
- To optimize the starvation timeout.
- To optimize run-length when the store is in critical state
- To fine tune the page open/close policy.

These settings are controlled by DDRCTRL_SCHED, DDRCTRL_SCHED1, DDRCTRL_HPR1, DDRCTRL_LPR1, DDRCTRL_WR1 registers.

8.5.3 Signal set 2: to monitor the refresh and self-refresh duration

The refresh commands are sent at regular intervals when the DDR is not in self-refresh. Monitoring the PowerDown commands sent when the DDR is idle, can be used to fine tune the low-power settings controlled by DDRCTRL_PWRCTRL and DDRCTRL_PWRTMG.

8.5.4 Signal set 3: to monitor the DDR stalling

By counting the duration where any DDRC store is in critical state, it is possible to fine tune the DDRCTRL scheduling and performance control.
The queue goes to critical state when the starvation counters time out

These settings are controlled by DDRCTRL_PCFGR/W, DDRCTRL_PCFGQOS, DDRCTRL_CFGWQOS registers for each port.

8.6 DDRPERFM registers

8.6.1 DDRPERFM control register (DDRPERFM_CTL)

Address offset: 0x000

Reset value: 0x0000 0000

Note: Write-only register. A read request returns all zeros.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP	START
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STOP:** stop all the counters.
0: writing a ‘0’ has no effect.
1: writing a ‘1’ stops all the counters
Writing a ‘1’ when the counters are stopped (DDRPERFM_STATUS.BUSY = 0) is ignored

Bit 0 **START:** Start counters which are enabled, the time counter (TCNT) is always enabled.
0: writing a ‘0’ has no effect.
1: writing a ‘1’ Starts counters which are enabledW
writing a ‘1’ when the counters are running (DDRPERFM_STATUS.BUSY = 1) is ignored

Caution: A simultaneous write of START and STOP bits at ‘1’ has an unpredictable effect.

8.6.2 DDRPERFM configurationl register (DDRPERFM_CFG)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEL[1:0]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN[3:0]			
												rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **SEL[1:0]:** select set of signals to be monitored (from 0 to 3) (see signal set description in *Table 27: DDRPERFM signal sets*) and counters to be enabled

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **EN[3:0]:** enable counter x (from 0 to 3)
0: the corresponding counter is disabled and does not increment
1: the corresponding counter is enabled and incremented on event at each clock

Note: The time counter (TCNT) is enabled by default.

Caution: The DDRPERFM_CFG can be modified **only** when the DDRPERFM is stopped.
A write to this register is ignored when DDRPERFM_STATUS.BUSY = 1

8.6.3 DDRPERFM status register (DDRPERFM_STATUS)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY
r															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVF[3:0]			
												r	r	r	r

Bit 31 **TOVF**: total counter overflow
0: no overflow
1: counter overflow

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **BUSY:** Busy Status
0: DDRPERFM is stopped or a counter is in overflow
1: DDRPERFM is counting (at least TCNT is counting)

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **COVF[3:0]:** Counter x Overflow (with x from 0 to 3)
0: no overflow
1: counter overflow

Note: When any counter is in overflow, the DDRPERFM is stopped with all the counters keeping their value, the DDRPERFM needs a restart after overflowing counter has been cleared.

The counter overflow is also visible also by reading DDRPERFM_CNTx = 0xFFFF_FFFF

8.6.4 DDRPERFM counter clear register (DDRPERFM_CCR)

Address offset: 0x00C

Note: Write-only register. A read request returns all zeros

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCLR[3:0]			
												w	w	w	w

Bit 31 **TCLR**: time counter clear
0: writing a ‘0’ has no effect
1: writing a ‘1’ clears the time counter

Bits 30:4 Reserved, must be kept at reset value.

Bits 3:0 **CCLR[3:0]:** counter x Clear (with x from 0 to 3)
0: writing a ‘0’ has no effect
1: writing a ‘1’ clears the corresponding event counter x (with x from 0 to 3)

Note: A write to this register is ignored when DDRPERFM_STATUS.BUSY = 1.

8.6.5 DDRPERFM interrupt enable register (DDRPERFM_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVFIE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OVFIE**: overflow interrupt enable
0: overflow interrupt disabled
1: overflow interrupt enabled

8.6.6 DDRPERFM interrupt status register (DDRPERFM_ISR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVFF
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OVFF**: overflow flag
This flag is set when one counter is in overflow

8.6.7 DDRPERFM interrupt clear register (DDRPERFM_ICR)

Address offset: 0x018

Note: Write-only register. A read request returns all zeros

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVF
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OVF**: overflow flag
0: no action
1: clears the overflow flag

Note: Following a counter overflow, the programming sequence must be:
1) Clear the counter(s) by writing '1' to DDRPERFM_CCR appropriate bits
2) Clear the interrupt by writing '1' to DDRPERFM_ICR.OVF

8.6.8 DDRPERFM time counter register (DDRPERFM_TCNT)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CNT[31:0]**: total time, this is number of DDR controller clocks elapsed while DDRPERFM has been running.

Caution: The time counter must only be read when DDRPERFM is stopped.
This register is read as zero when DDRPERFM_STATUS.BUSY=1.

8.6.9 DDRPERFM event counter x register (DDRPERFM_CNTx)

Address offset: 0x030 + x * 0x008 (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CNT[31:0]**: event counter value.

Each counter is incrementing when the corresponding signal from the selected set is asserted. See *Table 27: DDRPERFM signal sets* .

Caution: The counters must only be read when DDRPERFM is stopped.
These registers are read as zero when DDRPERFM_STATUS.BUSY=1.

8.6.10 DDRPERFM hardware configuration register (DDRPERFM_HWCFG)

Address offset: 0x3F0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NCNT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **NCNT[3:0]:** number of counters for this configuration (4)

8.6.11 DDRPERFM version register (DDRPERFM_VER)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision number.

Bits 3:0 **MINREV[3:0]:** Minor revision number.

8.6.12 DDRPERFM ID register (DDRPERFM_ID)

Address offset: 0x3F8

Reset value: 0x0014 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** DDRPERFM unique identification.

8.6.13 DDRPERFM magic ID register (DDRPERFM_SID)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** magic ID for automatic IP discovery.

8.6.14 DDRPERFM register map

Table 28. Power control register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DDRPERFM_CTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP	START
	Reset value																															0	0
0x004	DDRPERFM_CFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN[3:0]			
	Reset value															0	0													0	0	0	0
0x008	DDRPERFM_STATUS	TOVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVF[3:0]			
	Reset value	0															0													0	0	0	0
0x00C	DDRPERFM_CCR	TCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCLR[3:0]			
	Reset value	0																												0	0	0	0

Table 28. Power control register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	DDRPERFM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVFIE
	Reset value																																0
0x014	DDRPERFM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVFF
	Reset value																																0
0x018	DDRPERFM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVF
	Reset value																																0
0x01C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x020	DDRPERFM_TCNT	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024 - 0x02C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x030	DDRPERFM_CNT0	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x038	DDRPERFM_CNT1	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x040	DDRPERFM_CNT2	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x048	DDRPERFM_CNT3	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C - 0x3EC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x3F0	DDRPERFM_HWCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NCNT[3:0]			
	Reset value																													0	1	0	0
0x3F4	DDRPERFM_VER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	DDRPERFM_ID	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
0x3FC	DDRPERFM_SID	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

9 Power control (PWR)

The power control (PWR) provides an overview of the supply architecture of the different power domains, and the control of the supply configuration.

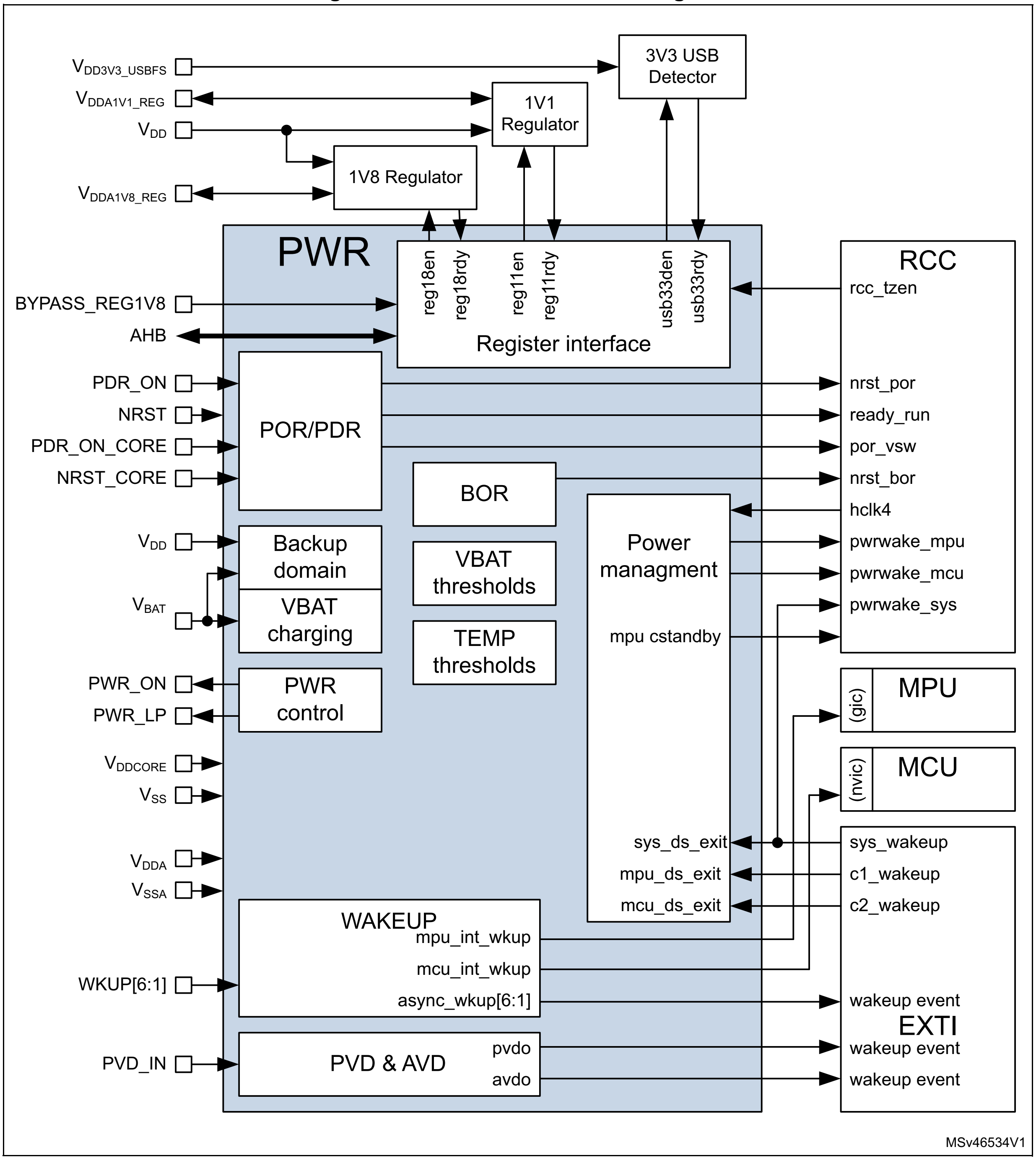
It provides an overview of the power supply supervisor features and provides the control of the V_{DDCORE} supply domain according to the operating modes. In the PWR section, the Cortex®- A7 is called MPU and the Cortex® -M4 is called MCU.

9.1 PWR main features

- System supply external regulation control
  - V_{DDCORE} supply request PWR_ON
  - V_{DDCORE} low power mode control PWR_LP
- Peripheral supply regulation
  - 1.8 V regulator (external to PWR IP)
  - 1.1 V regulator (external to PWR IP)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - VBAT thresholds
  - Temperature thresholds
  - V_{DDCORE} monitor
- VBAT charging
- Power management
  - Operating modes
  - Low-power modes
- Register access TrustZone® security

9.2 PWR block diagram

Figure 27. Power control block diagram



9.2.1 PWR pin overview

Table 29. PWR pin overview

Pin name	I/O	Description
V_{DD}	P	Main IO and VDD domain supply input
V_{DD_PLL}	P	PLLs supply input, shall be connected to V_{DD}(not used by PWR)
V_{BAT}	P	Backup battery supply input
V_{DDCORE}	P	Core domain supply
V_{DDA}	P	Analog domain supply
V_{DDA1V8_REG}	P	1V8 regulator supply output
V_{DDA1V1_REG}	P	1V1 regulator supply output
V_{DD3V3_USBFS}	P	USB FS IO 3V3 supply input
V_{DD3V3_USBHS}	P	USB HS PHY 3V3 supply input, shall be connected to V_{DD3V3_USBFS} (not used by PWR)
V_{DDQ_DDR}	P	DDR PHY supply input (not used by PWR)
V_{SS}	P	Main ground
V_{SSA}	P	Analog ground
V_{SS_USBHS}	P	USB HS PHY ground (not used by PWR)
AHB	I/O	AHB register interface
PWR_ON	O	Core supply enable output
PWR_LP	O	Core supply Low-power Stop mode selection output
PDR_ON	I	V_{DD} Power On Reset enable
NRST	IO	Application pad reset input/output
PDR_ON_CORE	I	V_{DDCORE} Power On Reset enable
NRST_CORE	I	V_{DDCORE} reset input (to be used when V_{DDCORE} Power On Reset is disabled)
BYPASS_REG1V8	I	1V8 regulator bypass
WKUP[6:1]	I	Wakeup inputs
PVD_IN	I	PVD external input
hclk4	I	PWR AHB bus clock input from RCC
pvdo	O	Programmable voltage detector output
avdo	O	Analog voltage detector output
nrst_por	O	V_{DD} Power-on reset
nrst_bor	O	V_{DD} Brownout reset
sys_ds_exit	I	System wakeup request from EXTI (sys_wakeup)
mpu_ds_exit	I	MPU wakeup request from EXTI (c1_wakeup)
mcu_ds_exit	I	MCU wakeup request from EXTI (c2_wakeup)
mpu_cstandby	O	MPU CStandby mode indication

Table 29. PWR pin overview (continued)

Pin name	I/O	Description
pwrwake_mpu	O	Clock wakeup request for MPU
pwrwake_mcu	O	Clock wakeup request for MCU
mpu_int_wkup	O	Wakeup interrupt to MPU
mcu_int_wkup	O	Wakeup interrupt to MCU
async_wkup[6:1]	O	Asynchronous wakeup to EXTI
rcc_tzen	I	TrustZone security enable (from TZEN register bit in RCC)

9.3 PWR power supplies

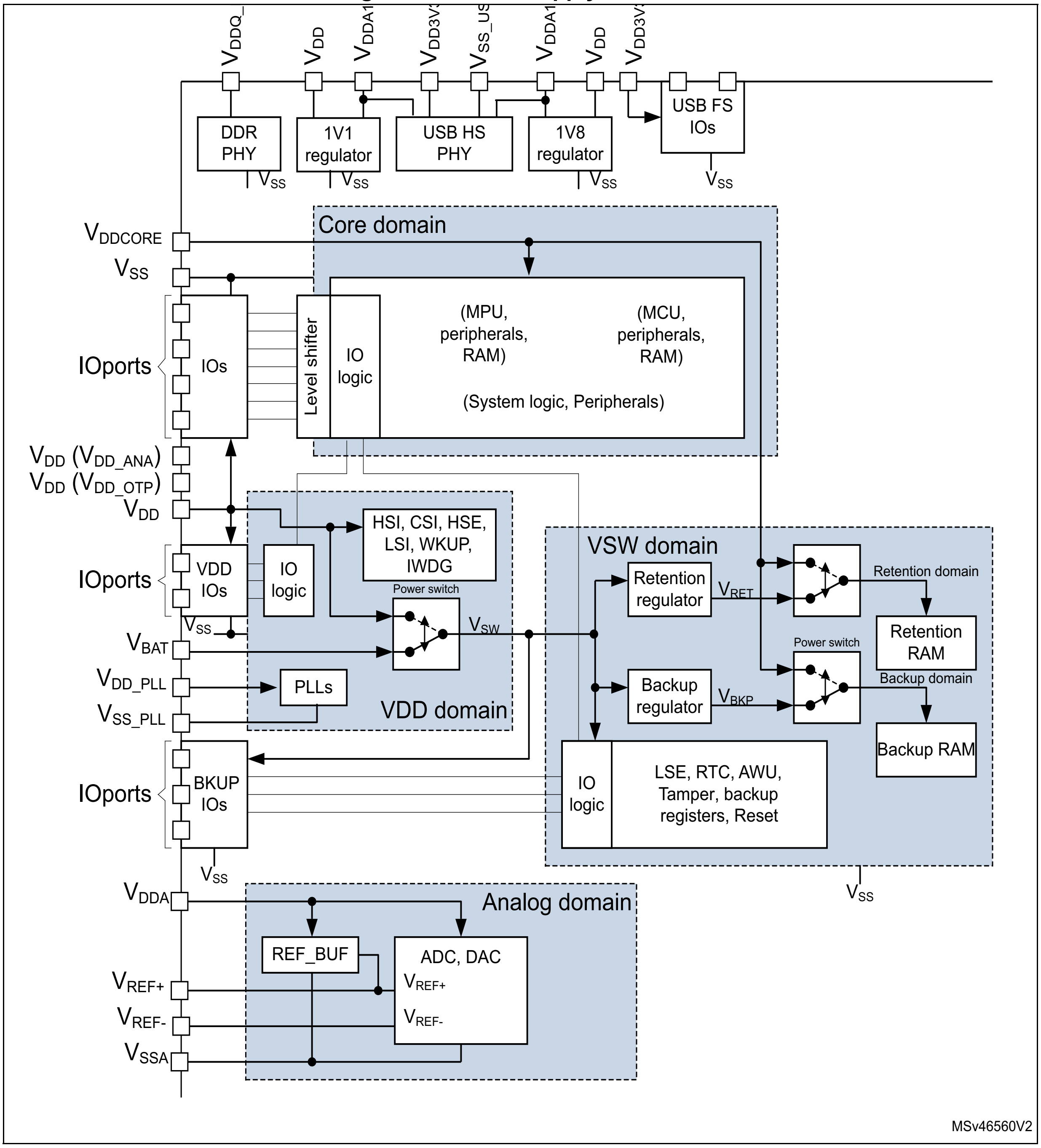
The system requires supply on V_{DD} and V_{DDCORE}, and provides independent supplies for V_{DDA}, V_{BAT}, V_{DD1V8_REG}, V_{DD1V1_REG}, V_{DD3V3_USBFS}, V_{DD3V3_USBHS} and V_{DDQ_DDR}. It provides regulated supplies for specific functions (1V8 regulator):

- V_{DD} power supply input for I/Os and system analog such as reset, power management and oscillators and PLLs.
- V_{DD_PLL} power supply input for PLLs, to be connected to the same supply as used for V_{DD}.
- V_{BAT} optional power supply input for backup domain when V_{DD} is not present (VBAT mode).
- V_{DDCORE} digital core domain supply, dependent on V_{DD} supply. VDD shall be present before V_{DDCORE}.
- V_{DDA} analog power supply input for ADCs, DACs and voltage reference buffers, independent from any other supply.
- V_{REF+} external reference voltage for ADC and DAC, independent from any other supply.
  - When the voltage reference buffer is enabled, supply output.
  - When the voltage reference buffer is disabled, independent external reference supply input.
- V_{SSA} separate analog and reference voltage ground.
- V_{DDA1V8_REG} 1V8 regulator supply, independent from any other supply.
  - When the 1V8 regulator is enabled, supply output.
  - When the 1V8 regulator is disabled, independent external supply input.
- V_{DDA1V1_REG} 1V1 regulator supply, independent from any other supply.
- V_{DD3V3_USBFS} supply input for USB FS IOs, independent from any other supply.
- V_{DD3V3_USBHS} supply input for USB HS PHY, to be connected to the same supply as used for V_{DD3V3_USBFS}.
- V_{SS_USBHS} separate USB HS PHY ground.
- V_{DDQ_DDR} supply input for DDR PHY, independent from any other supply.
- V_{SS} common ground for all supplies except for analog.

Note: Depending on the operating power supply range, some peripheral may be used with limited functionality and performance. For more details refer to section "General operating conditions" in the datasheet.

Note: An external DDR memory is to be supplied from an external regulator which is NOT controlled by the PWR. When the system enters low-power modes it is up to the firmware to control the DDR memory supply directly through the external regulator. When the system enters Standby mode the PWR_ON pin may be used by the external DDR memory regulator to power down the DDR memory supply.

Figure 28. Power supply overview



9.3.1 PWR core domain

The V_{DDCORE} core domain supply is provided from an external regulator. The V_{DDCORE} supplies all the digital circuitries except the backup domain and the standby circuitry when in Standby mode.

Following a V_{DD} power-on reset, the external regulator is enabled via PWR_ON and supplies the V_{DDCORE}.

The external regulator may operate in different modes (Main, Low-power, Off) which is used depending on the system operating modes (Run, Stop, LP-Stop, LPLV-Stop and Standby).

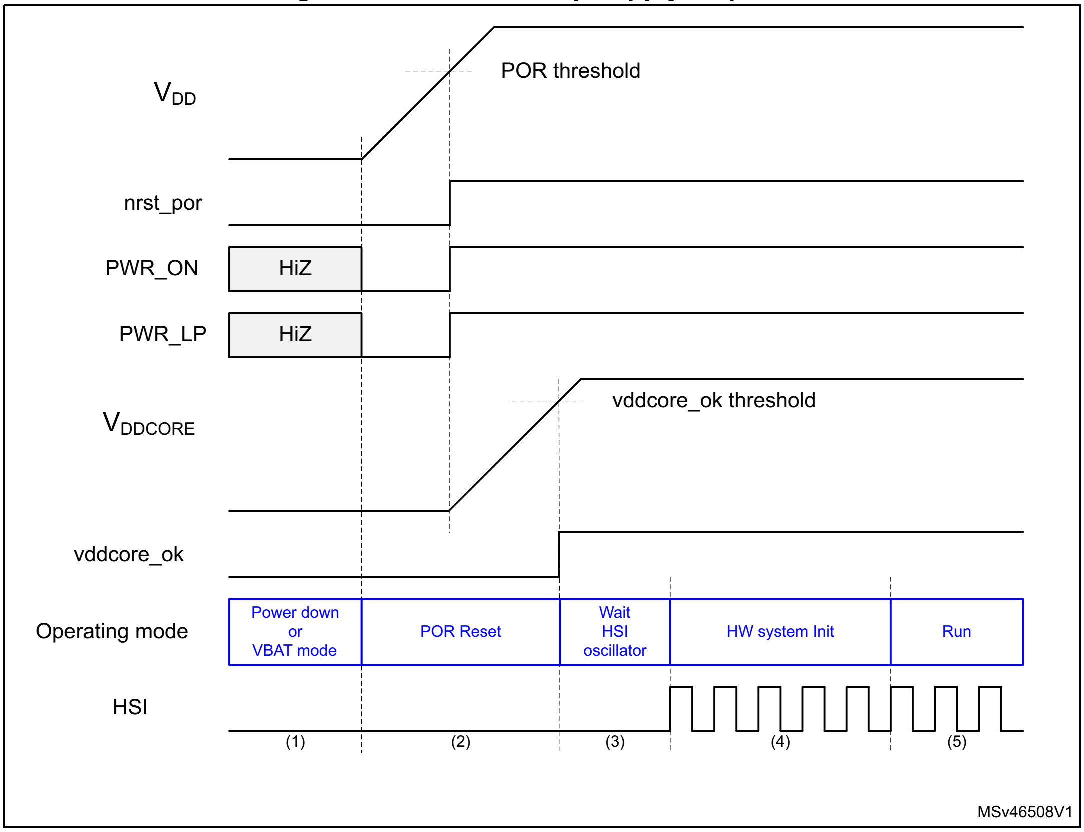
- In Run mode
  - The regulator supplies full power to the V_{DDCORE} domain (core, memories and digital peripherals), in its Main mode, both PWR_ON and PWR_LP are set high.
- In Stop mode
  - The regulator supplies the V_{DDCORE} domain to retain the content of registers and internal memories. The external regulator is kept in Main mode to allow a fast exit. The Low-power mode signaling is disabled with the LPDS register bit in *PWR control register 1 (PWR_CR1)*. Both PWR_ON and PWR_LP are kept set high.
- In LP-Stop mode
  - The regulator supplies the V_{DDCORE} domain to retain the content of registers and internal memories. The external regulator is allowed to enter Low-power mode. The Low-power mode signaling on PWR_LP is enabled with the LPDS register bit in *PWR control register 1 (PWR_CR1)*. The PWR_ON is kept set high or follows PWR_LP as selected in LPCFG in *PWR control register 1 (PWR_CR1)*, and PWR_LP is cleared to low.
- In LPLV-Stop mode
  - The regulator supplies the VDDCORE domain to retain the content of registers and internal memories. The external regulator is allowed to enter Low-power low-voltage mode. The Low-power mode signaling on PWR_LP is enabled with the LPDS register bit in *PWR control register 1 (PWR_CR1)*. The PWR_ON is kept set high or follows PWR_LP as selected in LPCFG in *PWR control register 1 (PWR_CR1)*, and PWR_LP is cleared to low. The Low-power Low-voltage mode is enabled by the LVDS register bit in *PWR control register 1 (PWR_CR1)*, the V_{DDCORE} supply level may be lowered during LPLV-Stop mode.
- In Standby mode
  - The PWR_ON and PWR_LP signals are both set low, the external regulator is switched off and the V_{DDCORE} domain is powered down. The content of the registers and memories are lost except for the backup domain and the Standby circuitry.

9.3.2 PWR supply system startup sequence

The system startup sequence is shown in *Figure 29*.

1. From power-on the POR monitors the V_{DD} supply. Once the supply is above the POR threshold level, the external Voltage regulator providing the V_{DDCORE} supply is enabled via the PWR_ON signal.
2. As long as the V_{DDCORE} is below the vddcore_ok threshold, the system is kept in reset.
3. Once the V_{DDCORE} supply is above the vddcore_ok threshold level, the system is taken out of reset, where after the HSI oscillator is enabled.
4. Once the HSI oscillator is stable, the hardware initializes the system (option bytes loading), then the Run mode is entered.

Figure 29. Device startup supply sequence

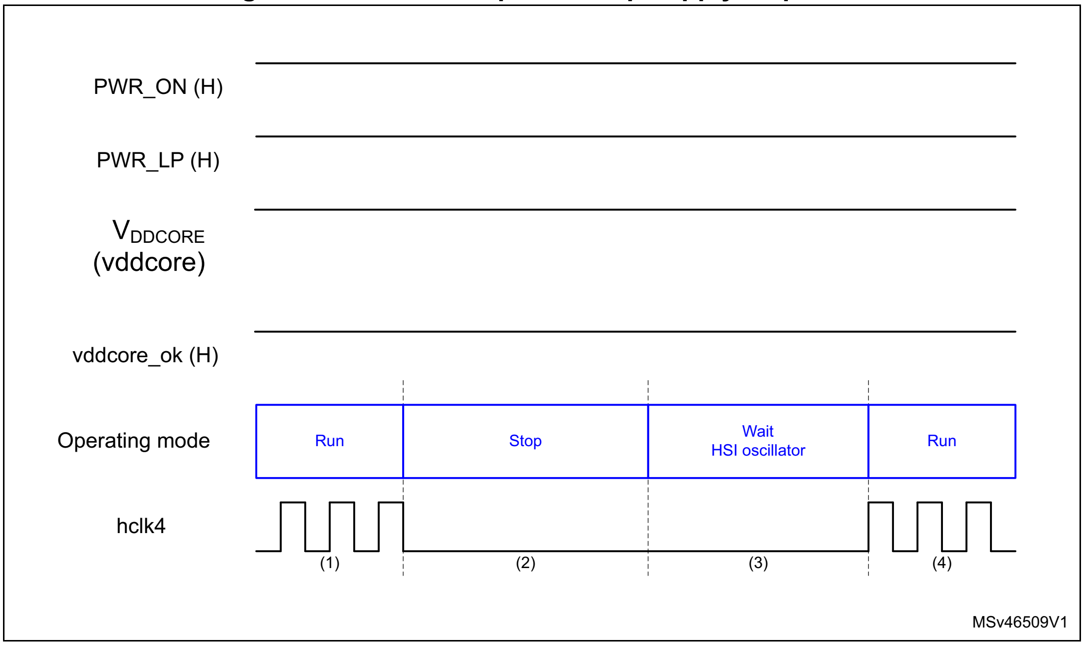


9.3.3 PWR supply Stop sequence

The system Stop mode sequence is shown in *Figure 30*. The Stop mode is entered when selected by the LPDS register bit in *PWR control register 1 (PWR_CR1)*.

1. When entering Stop mode, the hclk4 is stopped. The oscillators can be stopped in order to reduce the power consumption. For detailed oscillators status in Stop mode, see *Section 10.4.2: Oscillators description* and *Section 10.4.7: Clock generation in Stop and Standby modes*.
2. In Stop mode both the PWR_ON and PWR_LP are kept high. The external regulator remains in main mode.
3. On a wakeup event, the clock restore process is performed. See RCC: *The clock restore sequence description* for more details.
4. Once the oscillator is stable, the system enters into Run mode.

Figure 30. Device low-power Stop supply sequence

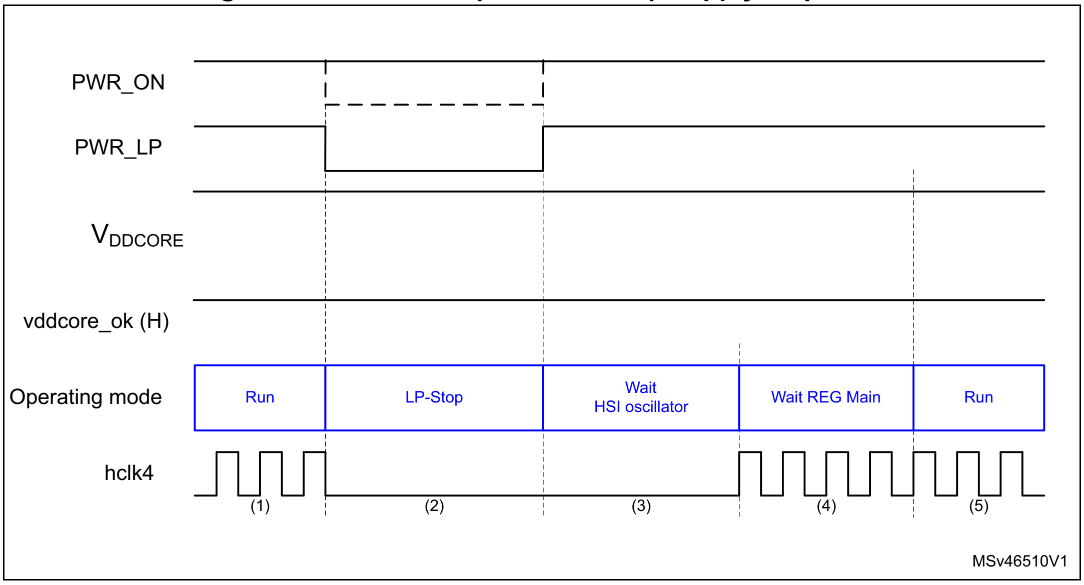


9.3.4 PWR supply LP-Stop sequence

The system LP-Stop mode sequence is shown in *Figure 31*. The LP-Stop mode is entered when selected by the LPDS register bit in *PWR control register 1 (PWR_CR1)*.

1. When entering LP-Stop mode the hclk4 is stopped. The oscillators can be stopped in order to reduce the power consumption. For detailed oscillators status in Stop mode, see *Section 10.4.2: Oscillators description* and *Section 10.4.7: Clock generation in Stop and Standby modes*
2. In the LP-Stop mode the PWR_ON is either kept high (pwr_on) or follows the PWR_LP (pwr_onlp) as selected with the LPCFG register bit in *PWR control register 1 (PWR_CR1)*. The PWR_LP is asserted, signaling LP-Stop. After, the external regulator may enter Low-power mode.
3. On a wakeup event the external regulator is taken out of Low-power mode via PWR_LP, and the clock restore process is performed. See RCC: *The clock restore sequence description* for more details.
4. Once the oscillator is stable, the PWRLP_TEMPO delay is timed out to allow the external regulator to switch to main mode (see *Section 10.7.92: RCC PWR_LP delay control register (RCC_PWRLPDLYCR)*).
5. After the delay timeout the system enters into Run mode.

Figure 31. Device low-power LP-Stop supply sequence

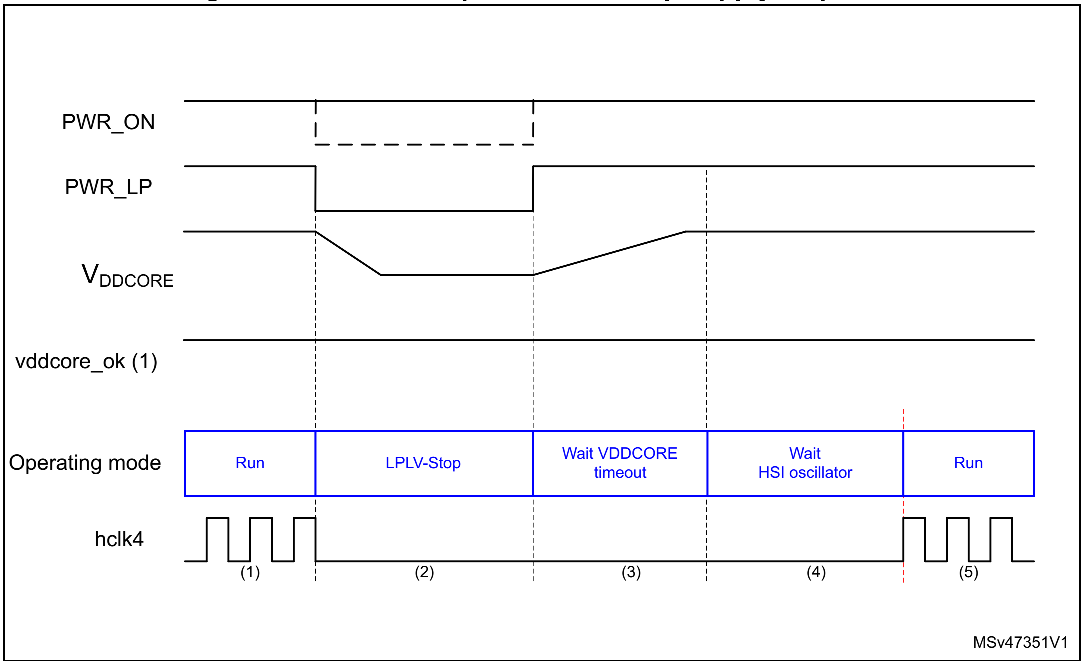


9.3.5 PWR supply LPLV-Stop sequence

The system LPLV-Stop mode sequence is shown in *Figure 32*. The LPLV-Stop mode is entered when selected by the LPDS and LVDS register bits in *PWR control register 1 (PWR_CR1)*.

1. When entering LPLV-Stop mode the hclk4 is stopped. The oscillators can be stopped in order to reduce the power consumption. For detailed oscillators status in Stop mode, see *Section 10.4.2: Oscillators description* and *Section 10.4.7: Clock generation in Stop and Standby modes*.
2. In the LPLV-Stop mode the PWR_ON is either kept high (pwr_on) or follows the PWR_LP (pwr_omlp) as selected with the LPCFG register bit in *PWR control register 1 (PWR_CR1)*. The PWR_LP is asserted, signaling LPLV-Stop. After, the external regulator may enter Low-power low-voltage mode and lower the V_{DDCORE} supply.
3. On a wakeup event the external regulator is taken out of Low-power low-voltage mode via PWR_LP and a delay $t_{SEL\_VDDCORETEMPO}$, is started to wait for the V_{DDCORE} to reach the Run mode operating supply level.
4. After this delay the HSI is enabled and the clock restore process is performed. See RCC: *The clock restore sequence description* for more details.
5. Once the oscillator is stable, the system enters Run mode.

Figure 32. Device low-power LPLV-Stop supply sequence

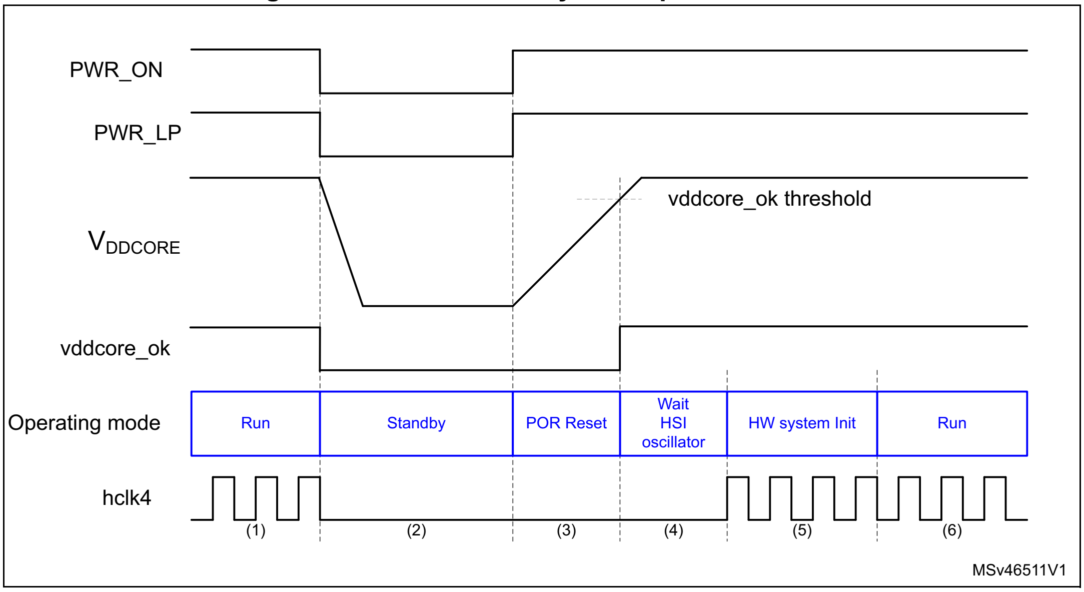


9.3.6 PWR supply Standby sequence

The system Standby mode sequence is shown in *Figure 33*.

1. When entering Standby mode, the PWR_ON signal requests the V_{DDCORE} supply to be powered off, the hclk4 is stopped. The oscillators are also stopped.
   If needed, a guaranteed minimum PWR_ON pulse low time can be defined in *PWR control register 3 (PWR_CR3)* bits POPL.
2. On a wakeup event, the external regulator is requested, via the PWR_ON signal, to switch on the V_{DDCORE} supply. As long as the V_{DDCORE} is below the vddcore_ok threshold, the system is kept in Standby.
   When a wakeup event occurs before the guaranteed minimum PWR_ON pulse low time, the external regulator is only requested to switch on the V_{DDCORE} supply till after the minimum PWR_ON pulse low time.
3. Once the V_{DDCORE} supply is above the vddcore_ok threshold level, the HSI oscillator is enabled.
4. Once the oscillator is stable, the hardware initializes the system (option bytes loading), after which Run mode is entered.

Figure 33. Device Standby mode power control.



In Standby mode when the V_{DDCORE} OK monitoring (vddcore_ok) is disabled by PDR_ON_CORE, the NRST_CORE signal shall be activated whenever V_{DDCORE} is powered down.

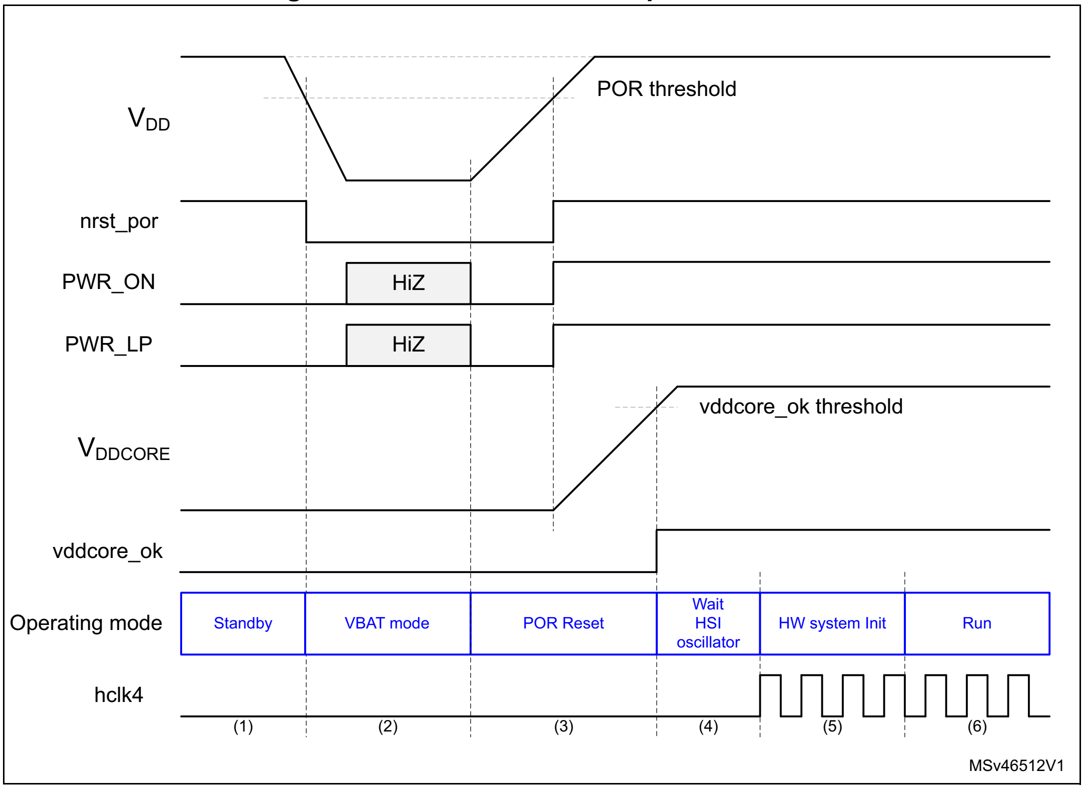
PDR_ON_CORE is reserved for STMicroelectronics production tests and must always be connected to V_{DD} in an application.

9.3.7 PWR supply VBAT sequence from Standby.

The system VBAT mode sequence after Standby is shown in *Figure 34*.

1. When in Standby mode, the device can be set in VBAT mode by powering down the V_{DD} supply. The pwr_on signal keeps requesting the V_{DDCORE} supply to be powered off.
2. When powering down V_{DD}, the device enters VBAT mode. The PWR_ON and PWR_LP signals become HiZ.
3. To exit VBAT mode, the V_{DD} supply shall be reestablished, the POR monitors the V_{DD} supply. Once the supply is above the POR threshold level the External Voltage regulator providing the V_{DDCORE} supply is enabled via the pwr_on signal:
   As long as the V_{DDCORE} is below the vddcore_ok threshold the system is kept in reset.
4. Once the V_{DDCORE} supply is above the vddcore_ok threshold level, the system is taken out of reset, where after the HSI oscillator is enabled.
5. Once the oscillator is stable, the hardware initializes the system (option bytes loading), after which Run mode is entered.

Figure 34. Device VBAT mode power control



When VBAT mode is entered from Run, Stop, LP-Stop or LPLV-Stop mode, the V_{DD} and V_{DDCORE} supplies shall both be powered down.

9.3.8 PWR V_{SW} domain

To retain the content of the V_{SW} domain (RTC, backup registers, backup RAM, and retention RAM) when V_{DD} is turned off, VBAT pin can be connected to an optional standby voltage supplied from a battery or an another source.

Switching to the V_{BAT} supply (VBAT mode) is controlled by the power-down reset (PDR) embedded in the reset block, monitoring the V_{DD} supply.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (refer to the datasheet for the value of $t_{RSTTEMPO}$) and V_{DD} > V_{BAT} + 0.6 V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

When the V_{DD} supply is present, the V_{SW} domain is supplied from V_{DD}. This allows to save V_{BAT} power supply battery life time.

If no external backup battery is used in the application, it is recommended to connect VBAT externally to V_{DD}.

Whenever the V_{DD} supply is present and above the PDR threshold, the backup domain I/Os are supplied by V_{DD}, the following functions are available:

- PC14 and PC15 can be used as either GPIO (if V_{DDCORE} is present) or as LSE pins.
- PC13, PI8 and PC1 can be used as either GPIO (if V_{DDCORE} is present) or as RTC or TAMP pins.

Note: Due to the fact that the VSW switch only sinks a limited amount of current, the use of PC1, PC13 to PC15, and PI8 as GPIOs is restricted. See Datasheet for details.

When the V_{DD} supply is absent and a supply is present on VBAT, the backup domain I/Os are supplied by V_{BAT}, the following functions are available in VBAT mode:

- PC14 and PC15 can be used as LSE pins only.
- PC13, PI8 and PC1 can be used as RTC or TAMP pins.

PWR V_{SW} domain access

After a system reset the RTC registers and RTC backup registers are protected against possible unwanted write accesses. To enable access to the V_{SW} domain, proceed as follows:

1. Set the DBP bit in the *PWR control register 1 (PWR_CR1)* to enable access to the V_{SW} domain.

For more detail on RTC and backup registers access see *Section 48: Real-time clock (RTC)* in reference manual.

PWR Backup RAM

The backup domain includes 4 Kbytes of backup RAM accessible in 32-bit, 16-bit or 8-bit data mode.

- In Run, Stop, LP-Stop and LPLV-Stop modes the backup RAM is supplied from the V_{DDCORE} supply.
- In Standby and VBAT modes, the backup RAM is supplied through the Backup regulator in the V_{SW} domain.

When the backup regulator is enabled by BREN register bit in *PWR control register 2 (PWR_CR2)*, the backup RAM content is retained even in Standby and/or VBAT mode (it can be considered as an internal EEPROM when V_{BAT} is always present). The backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or VBAT modes.

When enabling the backup regulator, the readiness of the backup supply shall be checked with the BRRDY register bit in *PWR control register 2 (PWR_CR2)*, before entering Standby or VBAT mode.

The backup RAM is not mass erased by an tamper event, instead it is read protected to prevent confidential data, such as cryptographic private key, from being accessed. To re-gain access to the backup RAM after a tamper event, it needs to be first erased. The backup RAM can be erased:

- After a tamper event, by performing a dummy write with zero as data to the backup RAM.

PWR retention RAM

The retention domain includes 64 Kbytes of retention RAM accessible in 32-bit, 16-bit or 8-bit data mode.

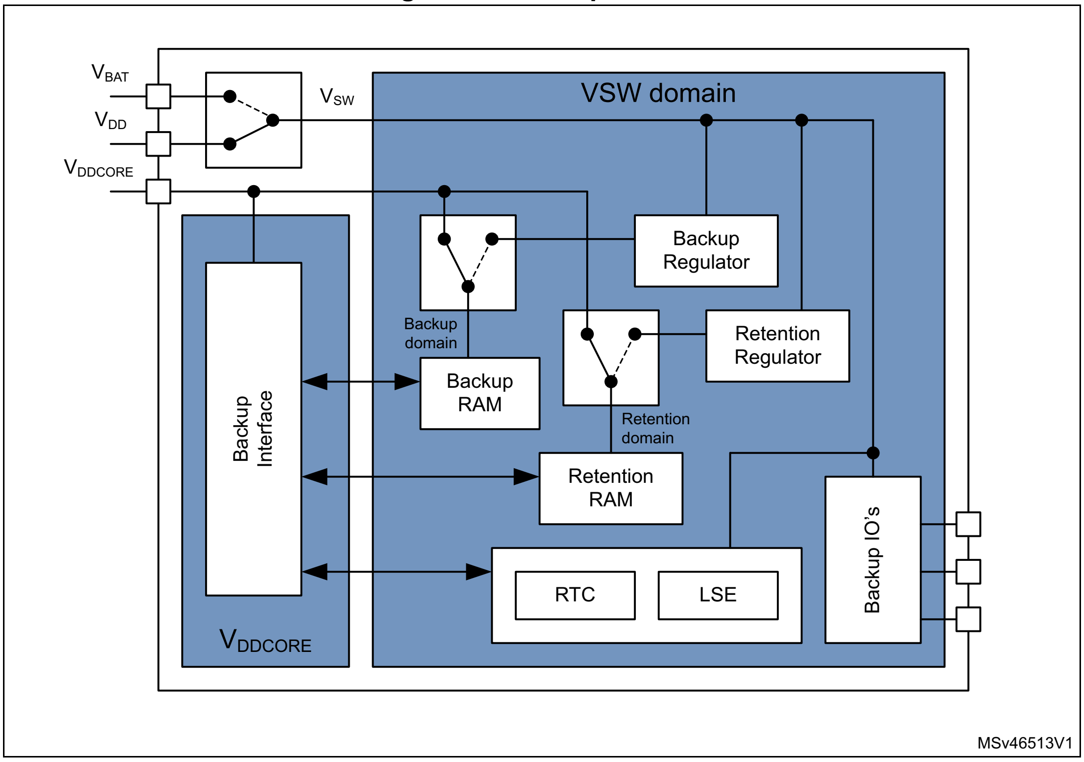
- In Run, Stop, LP-Stop and LPLV-Stop modes the retention RAM is supplied from the V_{DDCORE} supply.
- In Standby and VBAT modes, the retention RAM is supplied through the retention regulator in the V_{SW} domain.

When the retention regulator is enabled by the RREN register bit in *PWR control register 2 (PWR_CR2)*, the retention RAM content is retained even in Standby and/or VBAT mode (it can be considered as an Non-volatile memory when V_{BAT} is always present). The retention regulator can be ON or OFF depending whether the application needs the retention RAM function in Standby or VBAT modes.

When enabling the retention regulator, the readiness of the retention supply shall be checked with the RRRDY register bit in *PWR control register 2 (PWR_CR2)*, before entering Standby or VBAT mode.

In Standby and VBAT mode, when the retention supply drops below a threshold level, the RRRDY bit is reset. The retention regulator shall be disabled and then re-enabled to have the RRRDY bit set again. This function may be used to detect issues with the retention RAM integrity.

Figure 35. Backup domain



9.3.9 PWR VBAT battery charging

When V_{DD} is present, it is possible to charge the external battery on VBAT through an internal resistance between the 2 supplies.

The VBAT charging is done either through a 5 kOhm resistor, or through a 1.5 kOhm resistor depending on the VBRS bit value in *PWR control register 3 (PWR_CR3)*.

The battery charging is enabled by setting the VBE bit in *PWR control register 3 (PWR_CR3)*. It is automatically disabled when entering VBAT mode.

9.3.10 PWR analog supply

PWR separate analog V_{DDA} supply.

The analog supply domain has separate V_{DDA} and V_{SSA} pads to allow the supply to be filtered and shielded from noise on the PCB. This allows to improve the ADC and DAC conversion accuracy.

- The analog voltage supply input is available on a separate V_{DDA} pin.
- An isolated supply ground connection is provided on pin V_{SSA}.

PWR analog reference voltage V_{REF}.

To allow better accuracy on low voltage signals, the ADC and DAC have furthermore a separate reference voltage V_{REF+} pin. The user may connect a separate external reference voltage on V_{REF+}. The V_{REF+} controls the highest voltage, represented by the full scale value. The lower voltage reference V_{REF-} is connected together with V_{SSA}.

When enabled by the voltage reference buffer ENVR bit in the *Section 31.3.1: VREFBUF control and status register (VREFBUF_CSR)*, the V_{REF+} is provided from the internal voltage reference buffer. The internal voltage reference buffer can also provide a reference voltage to external components through the pin, see *Section 31: Voltage reference buffer (VREFBUF)*.

When the internal voltage reference buffer is disabled by ENVR, the V_{REF+} can also be used as a reference supply to external circuits.

9.3.11 PWR 1V8 regulator

The 1V8 regulator is supplied from the V_{DD} supply and generates the V_{DDA1V8_REG} supply. The USB HS PHY is directly connected to this supply.

The use of the integrated 1V8 regulator has to be enabled by the BYPASS_REG1V8 pin.

When enabled by the BYPASS_REG1V8 pin and the REG18EN register bit in *PWR control register 3 (PWR_CR3)*, the V_{DDA1V8_REG} is provided from the 1V8 regulator. Before using the V_{DDA1V8_REG}, the availability shall be checked with the REG18RDY register bit in *PWR control register 3 (PWR_CR3)*.

When the 1V8 regulator is disabled (bypass) by the BYPASS_REG1V8 pin or the REG18EN register bit, the V_{DDA1V8_REG} can be provided from an external supply.

For more information on the 1V8 regulator see *Section 9.3: PWR power supplies*.

9.3.12 PWR 1V1 regulator

The 1V1 regulator is supplied from the V_{DD} supply and generates the V_{DDA1V1_REG} supply. The USB HS PHY is directly connected to this supply.

The 1V1 regulator is enabled by the REG11EN register bit in *PWR control register 3 (PWR_CR3)*. Before using the supply from the 1V1 regulator, the availability shall be checked with the REG11RDY register bit in *PWR control register 3 (PWR_CR3)*.

For more information on the 1V1 regulator see *Section 9.3: PWR power supplies*.

9.4 PWR power supply supervision

The power supply level monitoring is available on the following supplies:

- V_{DD} via POR/PDR *Section 9.4.1*, BOR *Section 9.4.2*., and PVD monitor *Section 9.4.3*.
  - The POR/PDR monitoring flag is available from the PORRSTF register bit in the RCC.
  - The BOR monitoring flag is available from the BORRSTF register bit in the RCC.
  - The PVD monitoring flag is available from the PVDO register bit in *PWR control status register 1 (PWR_CSR1)*. In addition an interrupt and wakeup can be generated via the EXTI.
- V_{DDA} via AVD monitor, *Section 9.4.5*.
  - The AVD monitoring flag is available from the AVDO register bit in *PWR control status register 1 (PWR_CSR1)*. In addition an interrupt and wakeup can be generated via the EXTI.
- V_{BAT} via VBAT threshold, *Section 9.4.6*.
  - The VBAT monitoring could be enabled in *PWR control register 2 (PWR_CR2)*. The monitoring could generate tamper event via the TAMP.
- V_{DDCORE} via vddcore_ok which keeps V_{DDCORE} domain in reset as long as the level is not OK, *Section 9.5*.
  - The V_{DDCORE} OK monitoring flags are available from the SBF register bit in *PWR MCU control register (PWR_MCUCR)*, in *PWR MCU control register (PWR_MCUCR)* and from the VCORERSTF register bit in the RCC.
- V_{SW} via rst_vsw, which keeps V_{SW} domain in reset as long as the level is not OK.
  - The VSW monitoring has no flag. The VSW registers reset value could be used.
- V_{BKP} via a BRRDY in *PWR control register 2 (PWR_CR2)*
- V_{RET} via a RRRDY in *PWR control register 2 (PWR_CR2)*.
- V_{DD3V3_USBFS} via USB33RDY*PWR control register 3 (PWR_CR3)*.
- V_{DDA1V8_REG} via REG18RDY*PWR control register 3 (PWR_CR3)*
- V_{DDA1V1_REG} via REG11RDY*PWR control register 3 (PWR_CR3)*

9.4.1 PWR power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that allows a proper startup operation.

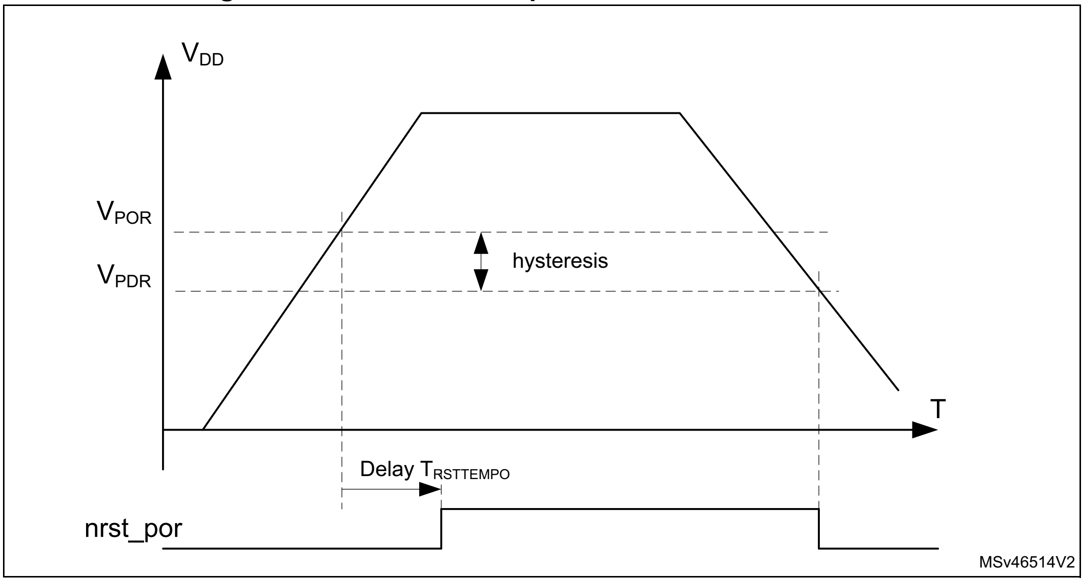
The system remains in Reset mode when V_{DD} is below a specified V_{POR} threshold, without the need for an external reset circuit. Once the V_{DD} supply level is above the V_{POR} threshold, the system is taken out of reset. More details concerning the power-on/power-down reset threshold, refer to the electrical characteristics in the datasheet.

The POR/PDR generates also an application reset (NRST).

The POR/PDR can be enabled/disabled by the device input pin PDR_ON. When disabled, the V_{DD} power on reset shall be provided on the NRST device input pin.

PDR_ON is reserved for STMicroelectronics production tests and must always be connected to V_{DD} in an application.

Figure 36. Power-on reset/power-down reset waveform



1. For the thresholds and hysteresis values, refer to the datasheet.

9.4.2 PWR brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the V_{DD} supply voltage reaches the specified V_{BOR} threshold.

The V_{BOR} threshold is configured through the system option bytes. By default, BOR is off. When selecting another BOR threshold, this takes effect only after the option bytes are loaded by the device. The following programmable V_{BOR} thresholds can be selected:

- BOR off (VBOR0): reset threshold level for full range voltage operation.
- BOR Level 1 (VBOR1): reset threshold level for above 2.10 V typ. voltage operation.
- BOR Level 2 (VBOR2): reset threshold level for above 2.40 V typ. voltage operation.
- BOR Level 3 (VBOR3): reset threshold level for above 2.70 typ. voltage operation.

Refer to Datasheet for more information on the BOR thresholds.

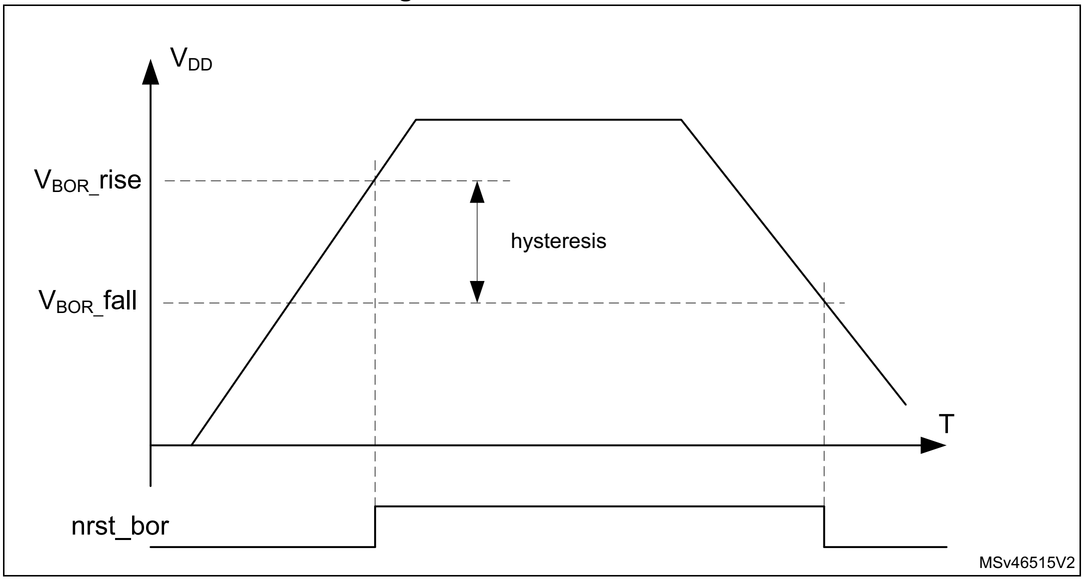
The BOR generates also an application reset (NRST).

When the BOR is enabled and the V_{DD} supply voltage drops below the selected V_{BOR} threshold, an Application reset (NRST) is generated.

The BOR can be disabled by programming the system option bytes. To disable the BOR function, V_{DD} must have been higher than V_{BOR0} to start the system option byte programming sequence. The power down is then monitored by the PDR (see *Section 9.4.1*)

The BOR threshold hysteresis for level 1, 2, and 3 is ~100 mV, for level 0 it is ~40 mV (between the rising and the falling edge of the supply voltage).

Figure 37. BOR thresholds



1. For the thresholds and hysteresis values, refer to the datasheet.

9.4.3 PWR VDDCORE OK reset

The system has an integrated circuitry that allows proper startup operation of the V_{DDCORE} domain.

The V_{DDCORE} domain remains in Reset mode when V_{DDCORE} is below the operation threshold vddcore_ok. Once the V_{DDCORE} supply level is above the operation threshold vddcore_ok, the V_{DDCORE} domain is taken out of reset. For more details concerning the V_{DDCORE} OK reset threshold, refer to the electrical characteristics of the datasheet.

The V_{DDCORE} OK detector can be enabled/disabled by the device input pin PDR_ON_CORE. When disabled, the V_{DDCORE} domain reset shall be provided on the NRST_CORE device input pin.

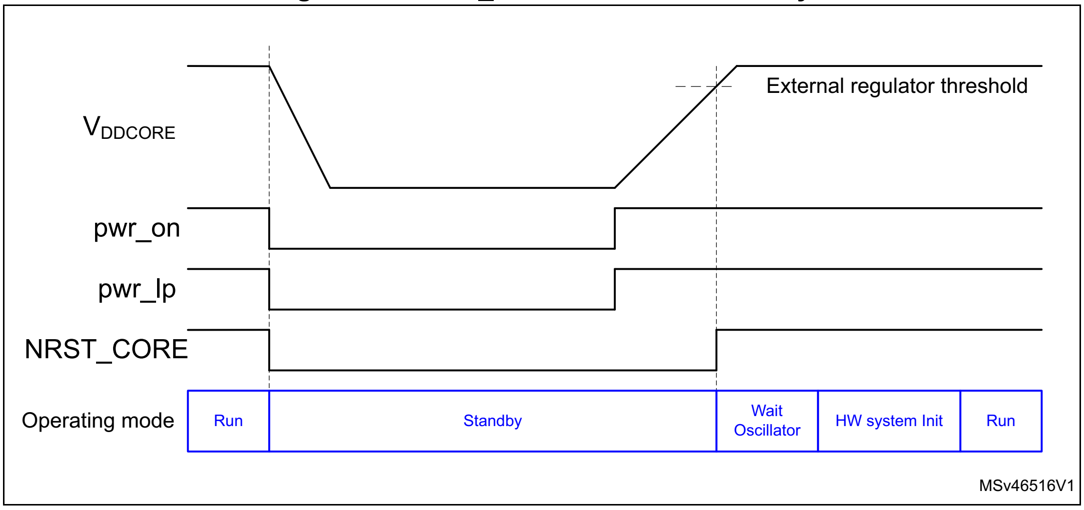
PDR_ON_CORE is reserved for STMicroelectronics production tests and must always be connected to V_{DD} in an application

When the V_{DDCORE} OK detector is disabled, the V_{DDCORE} supply level can be lowered in LPLV-Stop mode.

When the V_{DDCORE} OK detector is disabled, the NRST_CORE signal shall be used in Standby mode see *Figure 38*.

The NRST_CORE is operational irrespective of the PDR_ON_CORE state. Hence it also resets the V_{DDCORE} domain when the V_{DDCORE} OK detector is enabled by PDR_ON_CORE.

Figure 38. NRST_CORE control in Standby



9.4.4 PWR programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR control register 1 (PWR_CR1)*.

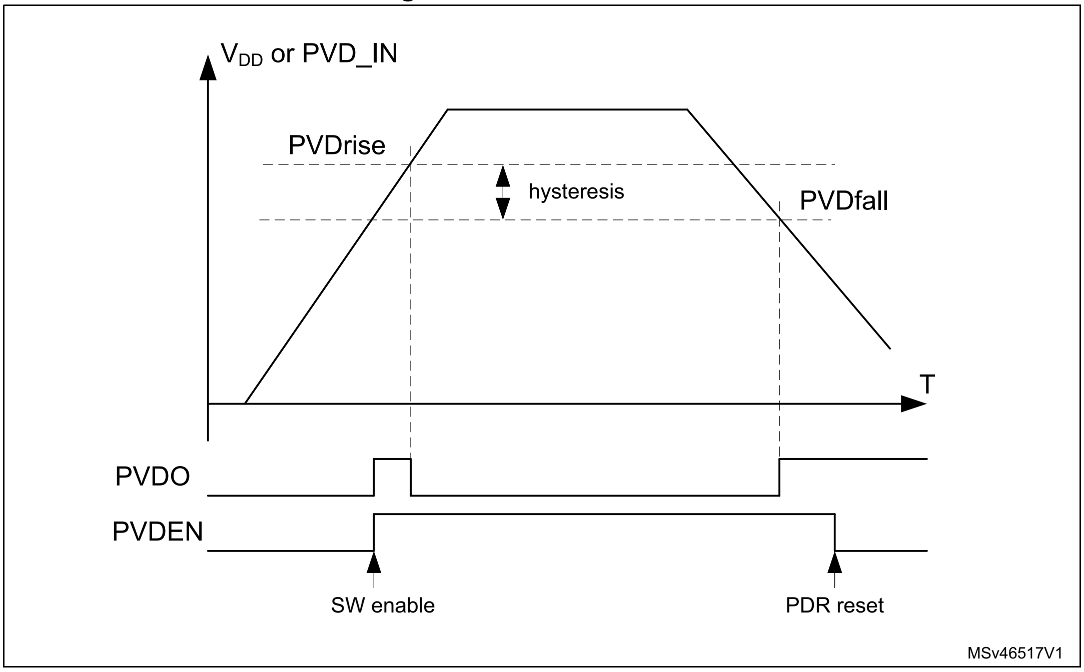
The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case, the voltage level on PVD_IN is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDEN bit in *PWR control register 1 (PWR_CR1)*.

The PVD is not available in Standby mode.

A PVDO flag is available, in the *PWR control status register 1 (PWR_CSR1)*, to indicate whether V_{DD} or voltage level on PVD_IN is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The PVDO output interrupt can be generated when V_{DD} or voltage level on PVD_IN drops below the PVD threshold and/or when V_{DD} or voltage level on PVD_IN rises above the PVD threshold depending on EXTI rising/falling edge configuration. As an example, the service routine could perform emergency shutdown tasks.

Figure 39. PVD thresholds



1. For the thresholds and hysteresis values, refer to the datasheet.

9.4.5 PWR analog voltage detector (AVD)

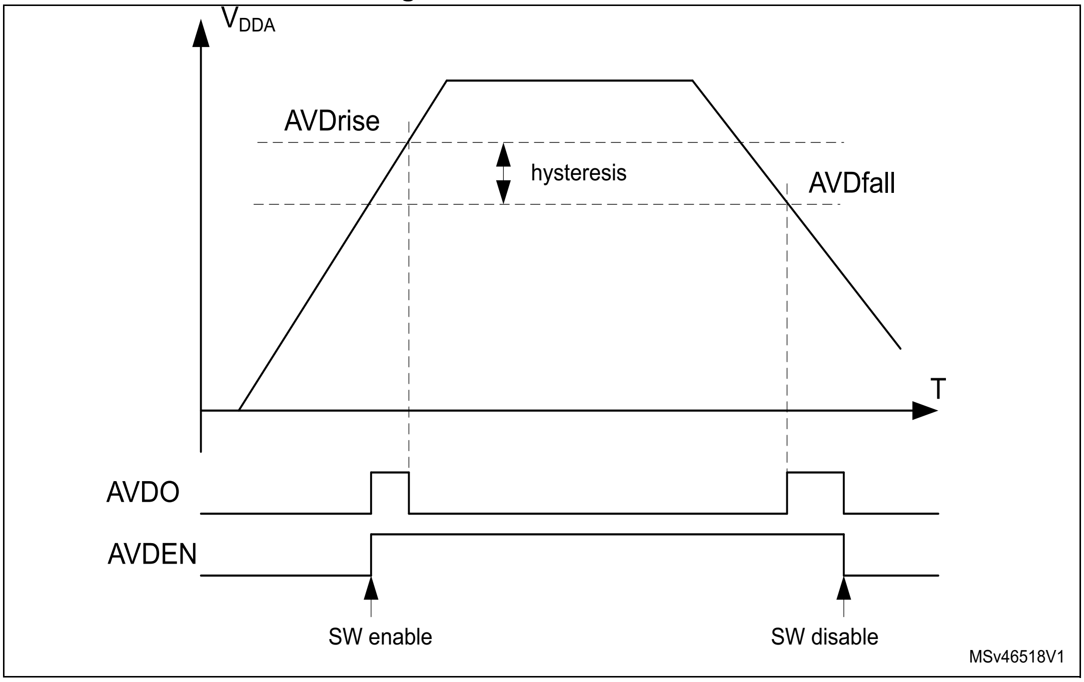
The AVD can be used to monitor the V_{DDA} supply by comparing it to a threshold selected by the ALS[1:0] bits in the *PWR control register 1 (PWR_CR1)*.

The AVD is enabled by setting the AVDEN bit in *PWR control register 1 (PWR_CR1)*.

The AVD is not available in Standby mode.

An AVDO flag is available, in the *PWR control status register 1 (PWR_CSR1)*, to indicate whether V_{DDA} is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when V_{DDA} drops below the AVD threshold and/or when V_{DDA} rises above the AVD threshold depending on EXTI rising/falling edge configuration. As an example, the service routine could indicate when the V_{DDA} supply drops below a minimum level.

Figure 40. AVD thresholds



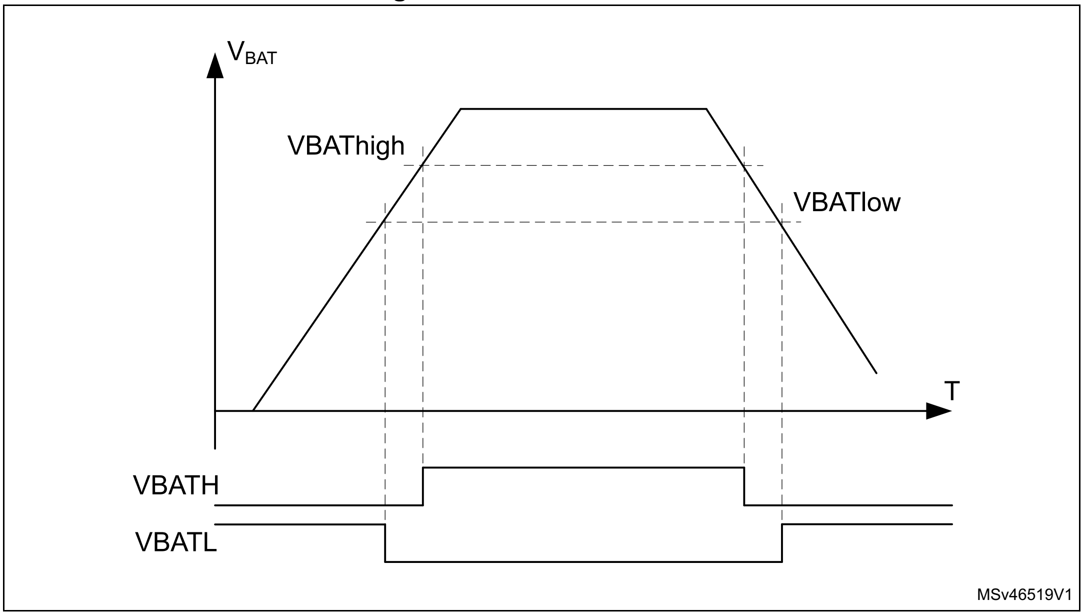
1. For the thresholds and hysteresis values, refer to the datasheet.

9.4.6 PWR battery voltage thresholds

During VBAT mode, the battery voltage V_{BAT} supply can be monitored by comparing it with two threshold levels. The VBAT supply monitor can be enabled/disabled with MONEN bit in *PWR control register 2 (PWR_CR2)*. When enabled, the battery voltage thresholds increases power consumption.

The VBATL and VBATH are available on tamper signals (see *Section 49: Tamper and backup registers (TAMP)*).

Figure 41. VBAT thresholds



1. For the thresholds and hysteresis values, refer to the datasheet.

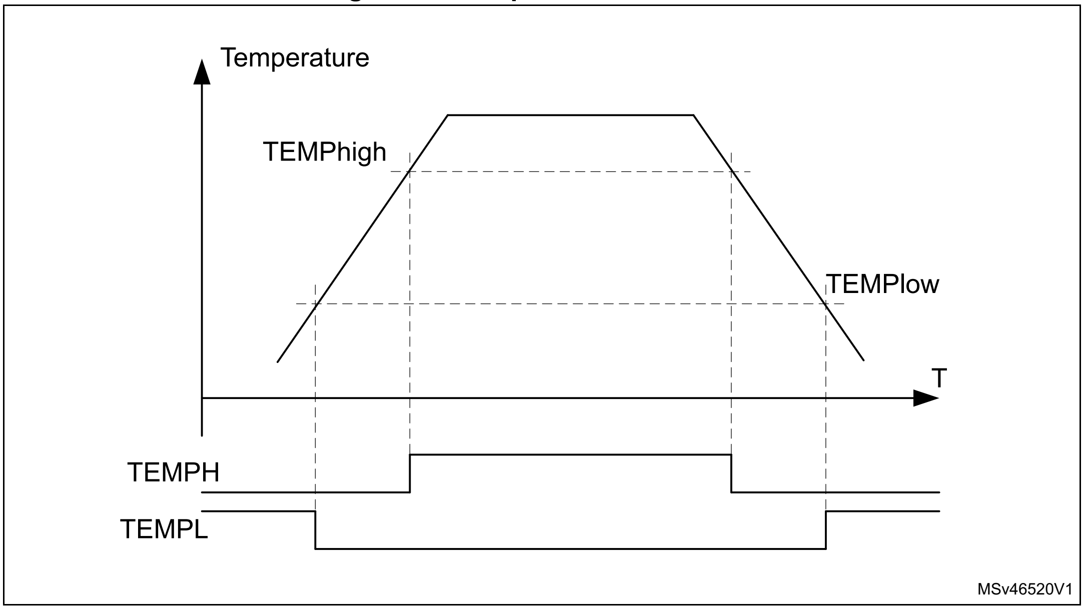
9.4.7 PWR temperature thresholds

The temperature can be monitored by comparing it with two threshold levels. TEMPH flag and TEMPL flag are available, in the *PWR control register 2 (PWR_CR2)*, to indicate if the device temperature is higher or lower than the threshold. The temperature monitor can be enabled/disabled with MONEN bit in *PWR control register 2 (PWR_CR2)*. When enabled, the temperature thresholds increases power consumption. As an example the levels could be used to trigger a routine to perform temperature control tasks.

The temperature thresholds, when enabled, are also available in Standby and VBAT modes.

The TEMPL and TEMPH are available on tamper signals (see *Section 48: Real-time clock (RTC)*).

Figure 42. Temperature thresholds



1. For the thresholds and hysteresis value, refer to the datasheet.

9.5 PWR power management

The power management controls the V_{DDCORE} supply in accordance with the system operation modes (see *Section 9.5.1: PWR operating modes*).

There is a single V_{DDCORE} domain for both the MPU and MCU systems. The system, MPU and MCU can be in one of the following operating modes:

- Run, CRun/CSleep (power On, clock On)
- Stop, LP-Stop, LPLV-Stop, CStop (power On, clock Off)
- CStandby (power On, clock Off, MPU reset).
- Standby (power Off, clock Off).

At the PWR controller level, the Dual core Cortex-A7 is seen as a single MPU.

The system operating mode depends on both MPU and MCU systems. For the system to be in Stop, LP-Stop, LPLV-Stop or Standby, both MPU and MCU need to be in CStop or CStandby, see *Table 33: System Low-power mode summary*.

The V_{DDCORE} is supplied from a single supply at a common level. The V_{DDCORE} supply level follows the system operating mode (Run, Stop, LP-Stop, LPLV-Stop and Standby).

9.5.1 PWR operating modes

Several system operating modes are available to tune the system according to the performance needs, i.e. when the MPU and MCU do not need to execute code and are waiting for an external event. It is up to the user to select the operating mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The operating modes allow the control of the clock distribution to the different system parts and the power of the system. The system operation mode is driven by the MPU sub-system and MCU sub-system. The MPU and MCU sub-systems may include multiple bus matrix depending on its peripheral allocation (see *Section 10: Reset and clock control (RCC)*). The following operating modes for the different system parts are defined:

- MPU and MCU sub-system modes:
  - **CRun**
    - The MPU/MCU and the MPU/MCU sub-system peripheral(s) according to RCC PERxEN are clocked.
  - **CSleep**:
    - The MPU/MCU clocks are stalled and the MPU/MCU sub-system allocated peripheral(s) clock operate according to RCC PERxLPEN. When the MPU/MCU enters CSleep with WFE, the MPU/MCU sub-system allocated peripheral(s) clock operate as in MPU/MCU CRun mode, irrespective of RCC PERxLPEN.
  - **CStop**:
    - The MPU/MCU and the MPU/MCU sub-system peripheral(s) clock are stalled. The wakeup depends on the system mode.
    1) When the system mode Run has been kept or Stop, LP-Stop or LPLV-Stop has been entered, the MPU/MCU wakes up with an interrupt or event.
    2) When the system mode Standby has been entered, the MPU/MCU wakes up with a reset.
  - **CStandby** (for MPU only)

    - The MPU and the MPU sub-system peripheral(s) clock are stalled, and the MPU always wakes up with a reset.

    1) When the system mode Run has been kept or Stop, LP-Stop or LPLV-Stop has been entered, the MPU wakes up with a local MPU reset, see *Section 10.3.3: The local resets*.

    2) When the system mode Standby has been entered, the MPU wakes up with a reset.
- System modes
  - **Run**
    The various sub-system clocks are running according to RCC configuration.
    - The MPU or MCU sub-system is in CRun or CSleep
  - **Stop**
    The LPDS bit in *PWR control register 1 (PWR_CR1)* selects Stop.
    The various sub-system clocks are stalled.
    - Both MPU and MCU sub-systems are in CStop or CStandby.
    and
    - At least one PDDS[a] bit selects Stop.
  - **LP-Stop**
    The LPDS bit in *PWR control register 1 (PWR_CR1)* selects LP-Stop. The LVDS bit in *PWR control register 1 (PWR_CR1)* selects normal voltage.

The various sub-system clocks are stalled.
- Both MPU and MCU sub-systems are in CStop or CStandby.
and
- At least one PDDS[a] bit select Stop.

– **LPLV-Stop**

  The LPDS and LVDS bits in *PWR control register 1 (PWR_CR1)* select LPLV-Stop.

  The various sub-system clocks are stalled.

  - Both MPU and MCU sub-systems are in CStop or CStandby

  and

  - At least one PDDS[a] bit selects Stop.

– **Standby**

  The system is powered down.
  - MPU sub-system is in CStandby or CStop with CSTBYDIS = 1 and MCU sub-system is in CStop
  and
  - All PDDS[a] bits select Standby.

In Run mode power consumption can be reduced by:

- Gating the clocks to the AHBx and APBx peripherals when they are unused with PERxEN (see *Section 10: Reset and clock control (RCC)*).

a. PPDS bit in the *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)* registers.

PWR MPU Low-power mode summary

The Dual Cortex-A7 core MPU system supports the following Low-power modes:

Table 30. MPU Low-power mode summary

<table>
<tr><th>MPU</th><th>Processor0</th><th>Processor1</th><th>L2 cache</th><th>Entry(1)</th><th>Wakeup</th><th>Processor0 clock</th><th>Processor1 clock</th><th>Bus matrix clocks</th></tr>
<tr><td rowspan="5">CRun</td><td>P0Run</td><td>P1Run</td><td rowspan="6">L2Run</td><td>-</td><td>-</td><td rowspan="3">ON</td><td>ON</td><td rowspan="6">ON</td></tr>
<tr><td rowspan="2">P0Run</td><td rowspan="2">P1Stop</td><td>WFE</td><td>A MPU Processor1 interrupt or event or reset</td><td rowspan="2">OFF</td></tr>
<tr><td>WFI</td><td>A MPU Processor1 interrupt or reset</td></tr>
<tr><td rowspan="2">P0Stop</td><td rowspan="2">P1Run</td><td>WFE</td><td>A MPU Processor0 interrupt or event or reset</td><td rowspan="7">OFF</td><td rowspan="2">ON</td></tr>
<tr><td>WFI</td><td>A MPU Processor0 interrupt or reset</td></tr>
<tr><td rowspan="2">CSleep</td><td rowspan="5">P0Stop</td><td rowspan="5">P1Stop</td><td>WFE(2)</td><td>A MPU(3) interrupt or event or reset</td><td rowspan="5">OFF</td></tr>
<tr><td rowspan="4">L2Stop</td><td>STPREQ_P[1:0] =not b'11
WFI(4)</td><td>A MPU(3) interrupt or reset</td><td>ON/OFF(5)</td></tr>
<tr><td rowspan="2">CStop</td><td>STPREQ_P[1:0] = b'11
MPU PDDS = 0
WFI(4)</td><td rowspan="2">A MPU(3) EXTI interrupt or reset</td><td rowspan="3">OFF/ON(6)</td></tr>
<tr><td>STPREQ_P[1:0] = b'11
MPU CSTBYDIS = 1
WFI(4)</td></tr>
<tr><td>CStandby</td><td>STPREQ_P[1:0] = b'11
MPU PDDS = 1
MPU CSTBYDIS = 0
WFI(4)</td><td>A MPU(3) EXTI interrupt or reset via a MPU reset.</td></tr>
</table>

1. For STPREQ_P[1:0] register bits functionality refer to the RCC.
2. At least one MPU processors is in WFE and the other MPU processor is in WFE or WFI.
3. Wakes up the respective processor associated with the event or interrupt.
4. All CPU processors are in WFI.
5. The MPU sub-system peripherals having a PERxLPEN bit operates accordingly.
6. The AXI bus clock may be ON due to the MCU with allocated peripherals activity.

Note: The Dual Cortex-A7 core MPU system does not support the individual processor shutdown mode from Arm.

PWR MCU Low-power mode summary

Table 31. MCU Low-power mode summary

MCU	Entry	Wakeup	Processor clock	Bus matrix clocks
CRun	-	-	ON	ON
CSleep	Arm Cortex-M4 SCR SLEEPDEEP = 0 WFE	Any MCU interrupt or event or reset	OFF	ON/OFF(1)
	Arm Cortex-M4 SCR SLEEPDEEP = 0 WFI or return from ISR	Any MCU interrupt or reset		
CStop	Arm Cortex-M4 SCR SLEEPDEEP = 1 WFE	Any MCU EXTI interrupt or event or reset		OFF/ON(2)
	Arm Cortex-M4 SCR SLEEPDEEP = 1 WFI or return from ISR	Any MCU EXTI interrupt or reset		

1. The MCU sub-system peripherals having a PERxLPEN bit operates accordingly.
2. The AHB bus clock may be ON due to the MPU activity.

PWR system Low-power modes summary

The system Low-power modes are derived from the MPU and MCU Low-power modes.

Table 32. PWR system mode summary

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="5">MPU</th></tr>
<tr><th>CRun</th><th>CSleep</th><th>CStop
PDDS = 0</th><th>CStop
PDDS = 1
CSTBYDIS = 1</th><th>CStandby</th></tr>
<tr><td rowspan="4">MCU</td><td>CRun</td><td>Run</td><td>Run</td><td>Run</td><td>Run</td><td>Run</td></tr>
<tr><td>CSleep</td><td>Run</td><td>Run</td><td>Run</td><td>Run</td><td>Run</td></tr>
<tr><td>CStop
PDDS = 0</td><td>Run</td><td>Run</td><td>Stop
LP-Stop
LPLV-Stop</td><td>Stop
LP-Stop
LPLV-Stop</td><td>Stop
LP-Stop
LPLV-Stop</td></tr>
<tr><td>CStop
PDDS = 1</td><td>Run</td><td>Run</td><td>Stop
LP-Stop
LPLV-Stop</td><td>Standby</td><td>Standby</td></tr>
</table>

Table 33. System Low-power mode summary

<table>
<tr><th rowspan="2">System</th><th rowspan="2">MPU</th><th rowspan="2">DDR[1]</th><th rowspan="2">MCU</th><th rowspan="2">Wakeup</th><th rowspan="2">Sys-oscillators[2]</th><th rowspan="2">hclk4</th><th rowspan="2">PWR_LP</th><th colspan="2">PWR_ON</th></tr>
<tr><th>LPCFG = 0</th><th>LPCFG = 1</th></tr>
<tr><td rowspan="3">Run</td><td>CRun or CSleep</td><td>Active / Auto refresh</td><td rowspan="2">CRun or CSleep</td><td rowspan="5">See Table 34, Table 31</td><td rowspan="3">ON</td><td rowspan="3">ON</td><td rowspan="4">1</td><td rowspan="5">1</td><td rowspan="4">1</td></tr>
<tr><td>CStop or CStandby</td><td>Auto refresh</td></tr>
<tr><td>CRun or CSleep</td><td>Active / Auto refresh</td><td>CStop</td></tr>
<tr><td>[3]
Stop
(LPDS = 0)</td><td rowspan="2">CStop or CStandby</td><td rowspan="2">Self refresh</td><td rowspan="2">CStop</td><td rowspan="2">ON/OFF[4]</td><td rowspan="3">OFF</td></tr>
<tr><td>[3]
LP-Stop,
LPLV-Stop
(LPDS = 1)</td><td>0
[5]</td><td>0 [5]</td></tr>
<tr><td>Standby</td><td>CStandby or
(CStop &
MPU PDDS = 1 & MPU
CSTBYDIS = 1)</td><td>Off / Self refresh</td><td>CStop &
MCU PDDS = 1</td><td>WKUP pins, RTC alarm, RTC wakeup, RTC tamper events, or Application reset (NRST, IWDG, ...)</td><td>OFF</td><td>0
[6]</td><td>0 [6]</td><td>0 [6]</td></tr>
</table>

1. The DDR operation state is only presented for information. i.e. The DDR may be Off when MPU is in CRun, e.g. when executing from BOOTROM or it may be in self refresh when the system is in any of its operation modes.
2. The sys-oscillators: HSI/HSE/CSI oscillators configured ON in RCC.
3. At least 1 PDDS bit selects the Stop mode (PPDS bit in registers *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)*).

4. When the system oscillator HSI, CSI or HSE is used, the state is controlled by respective xxxKERON, else the system oscillator is OFF (see *Section 10: Reset and clock control (RCC)*).
5. Settings in *PWR control register 3 (PWR_CR3)* bits POPL have no impact on the LP-Stop and LPLV-Stop mode PWR_ON and PWR_LP pulse low time.
6. A guaranteed minimum PWR_ON and PWR_LP pulse low time can be defined in *PWR control register 3 (PWR_CR3)* bits POPL.

Table 34. Functionalities depending on system operating mode(1)

Peripheral	Run	Sleep	Stop and LP-Stop		LPLV-Stop		Standby		VBAT
			-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
MPU	Y	-	R	-	R	-	-	-	-
MCU	Y	-	R	-	R	-	-	-	-
DBG	O	O	O	O	R	-	-	-	-
ROM memory	Y	Y	R	-	R	-	R	-	R
SYSRAM	Y(2)	O(2)	R	-	R	-	-	-	-
SRAMx (x = 1:3)	Y	O(2)	R	-	R	-	-	-	-
BKPSRAM	O(2)	O(2)	R	-	R	-	O(3)	-	-
RETSRAM	O(2)	O(2)	R	-	R	-	O(3)	-	-
DDRCTRL	O	O	R	-	R	-	-	-	-
QUADSPI	O	O	R	-	R	-	-	-	-
FMC	O	O	R	-	R	-	-	-	-
Backup Registers	Y	Y	R	-	R	-	R	-	R
Brown-out reset (BOR)	Y	Y	Y	Y	Y	Y	Y	Y	-
Programmable Voltage Detector (PVD)	O	O	O	O	O	O	-	-	-
Analog Voltage monitor (AVD)	O	O	O	O	O	O	-	-	-
VBATH/VBATL monitoring	-	-	-	-	-	-	-	-	O
TEMPH/TEMPL monitoring	O	O	O	O	O	O	O	O	O
MDMA	O	O	R	-	R	-	-	-	-
DMAx (x=1,2)	O	O	R	-	R	-	-	-	-
DMAMUX	O	O	R	-	R	-	-	-	-
High Speed Internal (HSI)	O	O	O(4)	-	-	-	-	-	-
High Speed External (HSE)	O	O	-	-	-	-	-	-	-
Low Speed Internal (LSI)	O	O	O	-	O	-	O	-	-
Low Speed External (LSE)	O	O	O	-	O	-	O	-	O
Multi-Speed Internal (CSI)	O	O	-	-	-	-	-	-	-
HSE CSS Clock Security System	O	O	-	-	-	-	-	-	-
LSE CSS Clock Security System	O	O	O	O	O	O	O	O	-
RTC / Auto wakeup	O	O	O	O	O	O	O	O	O
Number of Tamper pins	3	3	3	O	3	O	3	O	3
USBH, OTG	O	O	R	O	R	-	-	-	-

Table 34. Functionalities depending on system operating mode[1] (continued)

Peripheral	Run	Sleep	Stop and LP-Stop		LPLV-Stop		Standby		VBAT
			-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
SDMMCx (x=1:3)	O	O	R	-	R	-	-	-	-
FDCAN	O	O	R	-	R	-	-	-	-
CEC	O	O	R	O	R	-	-	-	-
ETH	O	O	R	O	R	-	-	-	-
MDIOS	O	O	O	O	R	-	-	-	-
USARTx (x = 1:8)	O	O	O[5]	O[5]	R	-	-	-	-
I2Cx (x = 1:6)	O	O	O[6]	O[6]	R	-	-	-	-
SPIx (x=1:6)	O	O	O[7]	O[7]	R	-	-	-	-
SAIx (x = 1:4)	O	O	R	-	R	-	-	-	-
DFSDM	O	O	R	-	R	-	-	-	-
DCMI	O	O	R	-	R	-	-	-	-
HDP	O	O	R	-	R	-	-	-	-
LTDC	O	O	R	-	R	-	-	-	-
DAC1-2	O	O	R	-	R	-	-	-	-
ADC1-2	O	O	R	-	R	-	-	-	-
VREFBUF	O	O	O	-	-	-	-	-	-
DTS	O	O	R	O	R	O	-	-	-
STGEN	O	O	R	-	R	-	-	-	-
TIMx (x = 1:17)	O	O	R	-	R	-	-	-	-
LPTIMx (x = 1:5)	O	O	O	O	R	O[8]	-	-	-
IWDGx (x=1,2)	O	O	O	O	O	O	O	O	-
WWDG	O	O	R	-	R	-	-	-	-
SysTick timer	O	O	R	-	R	-	-	-	-
RNGx (x = 1:2)	O	O	R	-	R	-	-	-	-
CRYPx[9] (x = 1:2)	O	O	R	-	R	-	-	-	-
HASHx (x = 1:2)	O	O	R	-	R	-	-	-	-
CRCx (x = 1:2)	O	O	R	-	R	-	-	-	-
IPCC	O	O	R	-	R	-	-	-	-

Table 34. Functionalities depending on system operating mode[1] (continued)

Peripheral	Run	Sleep	Stop and LP-Stop		LPLV-Stop		Standby		VBAT
			-	Wakeup capability	-	Wakeup capability	-	Wakeup capability	
HSEM	O	O	R	-	R	-	-	-	-
GPIOs	O	O	R	O	R	O	-	5 pins (10)	-

1. Legend: Y = Yes (Enable). O = Optional (Disable by default. Can be enabled by software). R = data/state retained. - = Not available; highlighted in gray for wakeup mode.
2. The SRAM clock can be gated on or off.
3. The BKPSRAM and RETSRAM content can optionally be retained when enabled.
4. Some peripherals with wakeup from Stop capability can request HSI to be enabled. In this case, HSI is woken up by the peripheral, and only feeds the peripheral which requested it. HSI is automatically put off when the peripheral does not need it anymore.
5. UART reception is functional in Stop mode, and generates a wakeup interrupt on Start, address match or received frame event.
6. I2C address detection is functional in Stop mode, and generates a wakeup interrupt in case of address match
7. SPI transmission and reception is functional in Stop mode, and generates a wakeup interrupt in case of FIFO thresholds, end of transfer, errors, ...
8. Only with LSE or LSI as kernel clock source.
9. Not present on all the product part numbers, see the related datasheet for more information.
10. The I/Os with wakeup from Standby/Shutdown capability is only available from a limited number of pins.

9.5.2 PWR power modes

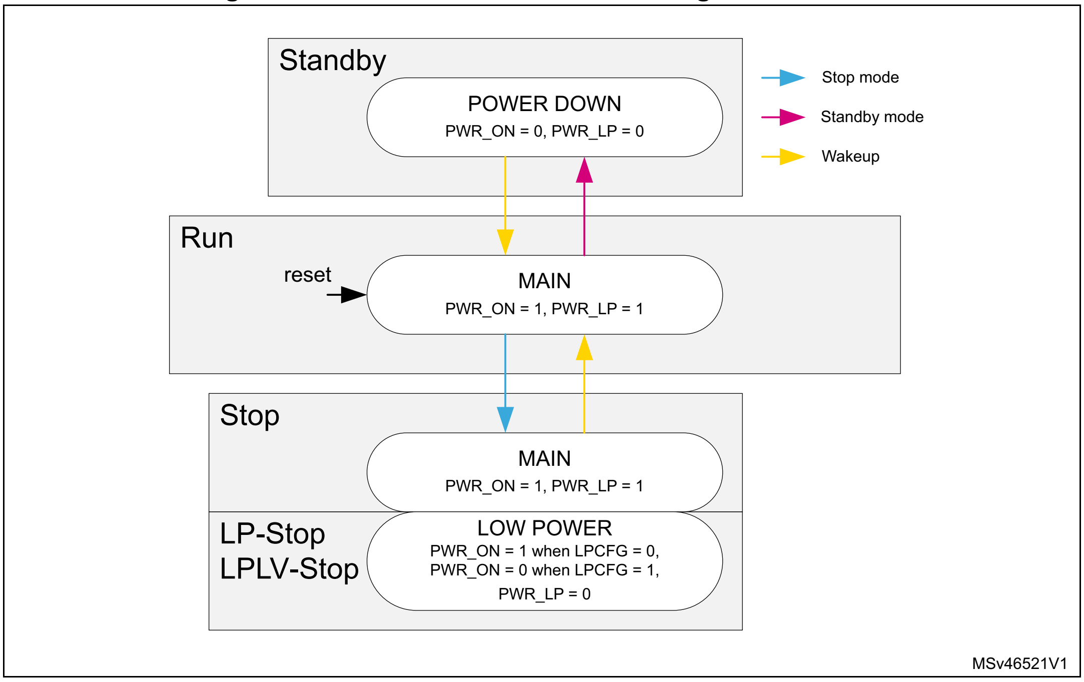
The power control handles the V_{DDCORE} supply for system Run, Stop, LP-Stop, LPLV-Stop and Standby modes. The system mode depends on the individual MPU and MCU sub-system modes (CRun, CSleep, CStop, CStandby).

- In Run mode the V_{DDCORE} is supplied.
- In Stop and LP-Stop modes the V_{DDCORE} is supplied.
- In LPLV-Stop mode the V_{DDCORE} may be supplied at a lower level.
- In Standby mode the V_{DDCORE} supply is switched off.

In system Stop mode, the PWR_LP is kept set high. In system LP-Stop mode, the PWR_LP is cleared to low, allowing an external regulator to switch between main power and Low-power mode with a reduced load capability and in LPLV-Stop mode to lower the V_{DDCORE} supply level.

In system Stop mode, the PWR_ON is kept asserted. In system LP-Stop mode, the PWR_ON is kept asserted or may be de-asserted according the register bit LPCFG in *PWR control register 1 (PWR_CR1)*, allowing an external regulator to switch between main power and Low-power mode with a reduced load capability and in LPLV-Stop mode to lower the V_{DDCORE} supply level. The definition (Power Off or Low-power mode) of the PWR_ON signal shall have be defined in the external regulator before entering LP-Stop or LPLV-Stop mode.

Figure 43. Power states and external regulator control

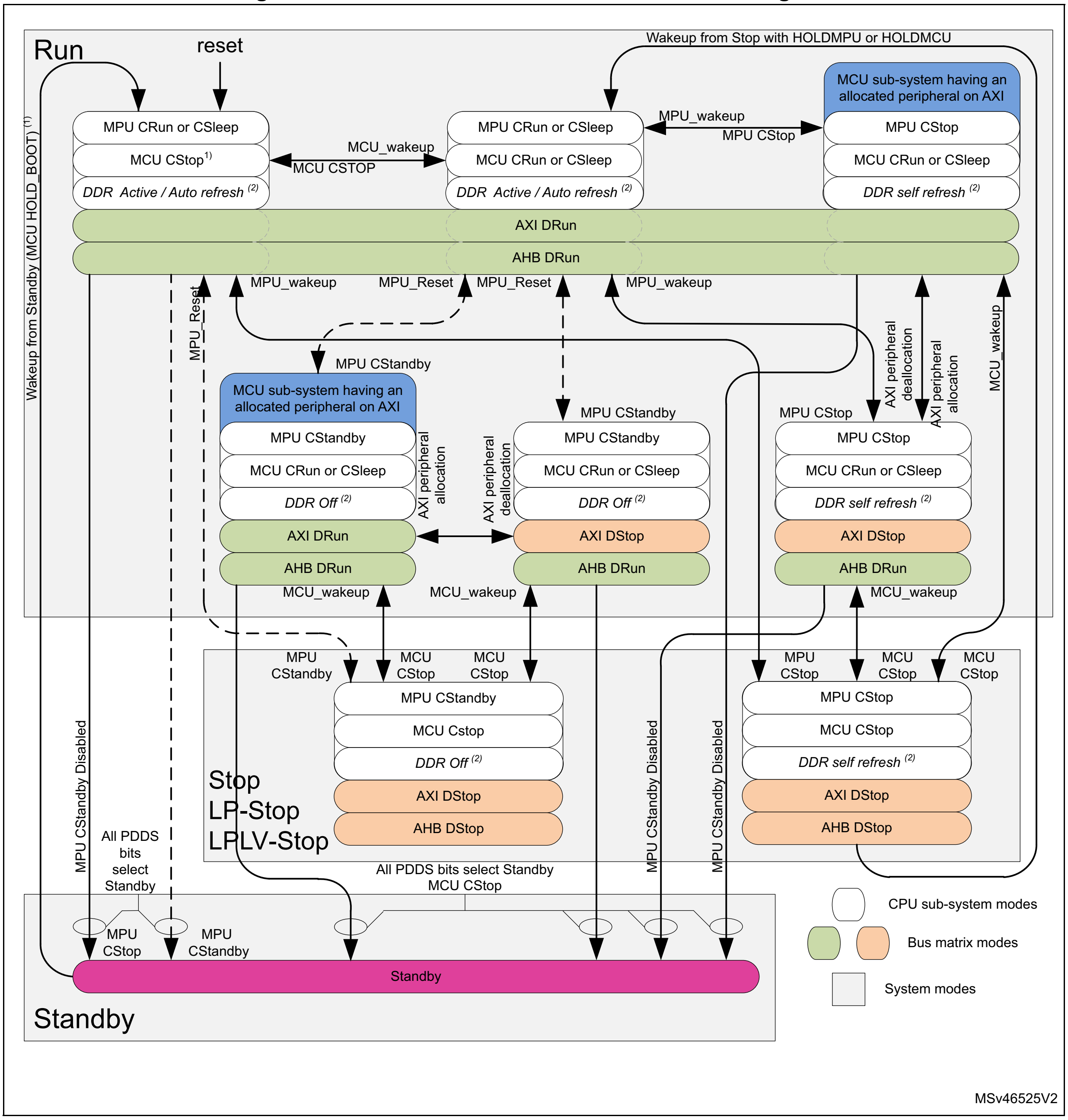


The system mode selection between Stop, LP-Stop or LPLV-Stop and Standby is controlled by the MPU PDDS /MCU PDDS bits and the MPU CSTBYDIS bits in *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)*. The system only enters Standby when both the MPU and MCU PDDS bits allow so.

Table 35. PDDS low-power mode control

MPU CSTBYDIS	MPU PDDS	MCU PDDS	System operation mode
x	0	x	Stop, LP-Stop, LPLV-Stop (with MPU in CStop)
0	1	0	Stop, LP-Stop, LPLV-Stop (with MPU in CStandby)
0	1	1	Standby (with MPU in CStandby)
1	1	1	Standby (with MPU in CStop)

Figure 44. Power control modes detailed state diagram



1. After a wakeup from Standby the MPU and MCU boot as described in *Section 10: Reset and clock control (RCC)* processor hold from standby. When ever MCU is in HOLD_BOOT it does not prevent the system to enter Stop, LP-Stop, LPLV-Stop or Standby.
2. The DDR operation state is only presented for information. i.e. The DDR may be Off when MPU is in CRun, e.g. when executing from Boot ROM.

Power control state transitions are initiated by:

- MPU or MCU going to CStop.
  - *Figure 44* Green transitions system stays in Run.
  - *Figure 44* Blue transitions system enters to Stop, LP-Stop or LPLV-Stop.
  - *Figure 44* Red transitions system enters to Standby.
- MPU going to CStandby
  - *Figure 44* Dotted Red line transitions, entered system mode depend also on MCU mode.
- EXTI wakeup event.
  - *Figure 44* Green transitions MPU or MCU wakes up as from CSleep.
  - *Figure 44* Blue transitions MPU or MCU wakes up from CStop. The STOPF is set.
- Wakeup event
  - *Figure 44* Dotted Red line transitions MPU wakes up from CStandby. The MPU SBFMPU is set.
    - When the system remained in Run no additional flags are set.
    - When the system wakes up from Stop, LP-Stop or LPLV-Stop additionally the STOPF is set.
  - *Figure 44* Red transitions MPU and MCU wakes up from Standby. The SBF is set.
- Peripheral allocation or deallocation
  - *Figure 44* Orange transitions MPU is in CStop or CStandby and MCU allocates a first or deallocates the last peripheral on the AXI bus.

The flags provided to determine from which mode the system exits are shown in *Table 34*. The MPU and MCU have their own set of flags which can be read from *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)*. The MPU and MCU have their own register reset bit CSSF to clear it's own wakeup flags. The MPU and MCU CSSF register bit is located in the same register as the MPU and MCU flags:

Table 36. Low-power mode exit flags

System mode	MPU mode	MCU mode	SBF	SBFMPU	STOPF	Comment
Run	CRun / CStop	CRun / CStop	0	0	0	System contents retained
	CStandby		0	1	0	
Stop LP-Stop LPLV-Stop	CStop	CStop	0	0	1	System contents retained, clock system stopped.
	CStandby		0	1	1	
Standby	CStandby		1	1	0	System contents lost
VBAT(1)	n.a.	n.a.	0	0	0	System contents lost

1. Exit from VBAT mode may be derived from the retained registers in the V_{SW} domain, which are not reset.

9.5.3 PWR system wakeup

From an application reset (NRST, IWDG, ...) and a wakeup from Standby, the system wakes up the MPU and the MCU is kept in HOLD. From this point the MPU BOOTROM manages the startup (see *Processor hold from Standby* in *Section 10: Reset and clock control*

(RCC)). From Stop, LP-Stop and LPLV-Stop the system can be configured to wake up the MPU or MCU independently.

9.6 PWR low-power modes

Several Low-power modes are available to save power when the MPU and/or MCU do not need to execute code (i.e. when waiting for an external event). It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

- Slowing down the CPU clock and/or the bus matrix clocks (see *Section 10: Reset and clock control (RCC)*)
- Controlling individual peripheral clocks (see *Section 10: Reset and clock control (RCC)*)
- Low-power modes
  - CSleep (MPU/MCU clock stopped)
  - CStop (MPU/MCU sub-system clock stopped)
  - CStandby (MPU sub-system clock stopped and wakeup via reset)
  - Stop (bus matrix clocks stalled, the oscillators can be stopped, for detailed oscillators status in Stop mode, see *Section 10.4.2: Oscillators description* and *Section 10.4.7: Clock generation in Stop and Standby modes*, normal mode signaled to external regulator).
  - LP-Stop and LPLV-Stop (bus matrix clocks stalled, the oscillators can be stopped, for detailed oscillators status in Stop mode, see *Section 10.4.2: Oscillators description* and *Section 10.4.7: Clock generation in Stop and Standby modes*, Low-power mode signaled to external regulator).
  - Standby (System powered down)

9.6.1 PWR slowing down the CPU clocks and/or the bus matrix clocks

In Run mode, the speed of the MPU, MCU and/or the bus matrix clocks can be reduced, for more details refer to *Section 10: Reset and clock control (RCC)*.

9.6.2 PWR controlling peripheral clocks

In Run mode, the bus clock for individual peripherals can be stopped by the RCC PERxEN register bits at any time to reduce power consumption.

To reduce power consumption in CSleep mode the individual peripheral clocks can be disabled by the RCC PERxLPEN register bit. For the peripherals still receiving a clock in CSleep mode, their clock can be slowed down before entering to CSleep mode.

9.6.3 PWR entering to low-power modes

MPU sub-system Low-power modes CSleep, CStop and CStandby are entered by the MPU when executing the WFI (Wait For Interrupt), or WFE (Wait for Event) instructions.

MCU sub-system Low-power modes CSleep and CStop are entered by the MCU when executing the WFI (Wait For Interrupt), or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M4 System Control register is set on Return from ISR.

The system may enter Stop, LP-Stop, LPLV-Stop or Standby when all EXTI wakeup sources are cleared and both the MPU and MCU are in CStop or CStandby.

9.6.4 PWR exiting from low-power modes

From CSleep mode the MPU and MCU sub-systems exit from low-power mode through any interrupt or event depending how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter to low-power mode, any peripheral interrupt acknowledged by the GIC/NVIC can wake up the system.
- If the WFE instruction is used to enter to low-power mode, the MPU and MCU exits from low-power mode as soon as an event occurs. The wakeup event can be generated either by:
  - GIC/NVIC IRQ interrupt.
    - For MPU, by enabling an interrupt in the peripheral control register and in the GIC.
    Only GIC interrupts with sufficient priority wake up and interrupt the MPU.
    - For MCU, when SEVONPEND = 0 in the Cortex®-M4 System Control register.
    By enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
    Only NVIC interrupts with sufficient priority wake up and interrupt the MCU.
    - For MCU when SEVONPEND = 1 in the Cortex®-M4 System Control register.
    By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.
    All NVIC interrupts wake up the MCU, even the disabled ones.
    Only enabled NVIC interrupts with sufficient priority wake up and interrupt the MCU.
  - Event
    - For MPU and MCU configuring an EXTI line in event mode. When the MPU or MCU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It may be necessary to clear the interrupt flag in the peripheral.

From CStop, Stop, LP-Stop and LPLV-Stop modes the MPU and/or MCU sub-systems exits from low-power mode only through an enabled EXTI interrupt or event depending on how the low-power mode was entered (see above).

From CStandby mode, when the system is in Run, Stop, LP-Stop or LPLV-Stop the MPU sub-system exits from low-power mode only through an enabled EXTI interrupt. Program execution restarts with a MPU reset.

From Standby mode the MPU sub-system exits from low-power mode only through an application reset (NRST, IWDG, ...), WKUPx pins event or a RTC event. Program execution restarts in the same way as after a power on reset (option bytes loading, reset vector fetched, etc.). When the system exits Standby mode the MCU is in HOLD. From this point the MPU BOOTROM manages the startup (see *Processor hold from Standby* in *Section 10: Reset and clock control (RCC)*).

9.6.5 PWR CSleep mode

The CSleep mode applies to the MPU and/or MCU sub-systems. In CSleep mode the MPU/MCU clock is stopped. The MPU and/or MCU sub-systems peripherals clocks operate according to the PERxLPEN bits.

PWR entering to CSleep mode

The CSleep mode is entered according to *Section 9.6.3: PWR entering to low-power modes*.

- For MPU one of the STPREQ_P[1:0] register bits in (see *Section 10: Reset and clock control (RCC)*) is cleared.
- For MCU the SLEEPDEEP bit in the Cortex®-M4 System Control register is cleared.

Refer to *Table 37* for MPU and *Table 38* for MCU, for details on how to enter to CSleep mode.

PWR exiting from CSleep mode

The CSleep mode is exited according to *Section 9.6.4: PWR exiting from low-power modes*.

Refer to *Table 37* for MPU and *Table 38* for MCU, for more details on how to exit from CSleep mode.

Table 37. MPU CSleep

CSleep mode	Description
Mode entry	WFI (Wait for Interrupt) while: – STPREQ_P[1:0] = not ‘b11 (see *Section 10: Reset and clock control (RCC)*). – MPU GIC interrupts and events cleared.
	WFE (Wait for Event) while: – MPU GIC interrupts and events cleared.
Mode exit	If WFI was used for entry: – Any Interrupt enabled in GIC: Refer to *Section 20.2: GIC Interrupts* If WFE was used for entry: – Any Interrupt even when disabled in GIC: refer to *Section 20.2: GIC Interrupts* or any event: refer to *Section 9.6.4: PWR exiting from low-power modes*
Wakeup latency	None

Table 38. MCU CSleep

CSleep mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 (Refer to the Cortex®-M4 System Control register.) – MCU NVIC interrupts and events cleared.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 (Refer to the Cortex®-M4 System Control register.) – MCU NVIC interrupts and events cleared.

Table 38. MCU CSleep (continued)

CSleep mode	Description
Mode exit	If WFI or return from ISR was used for entry: – Any Interrupt enabled in NVIC: Refer to *Section 20.2: GIC Interrupts* If WFE was used for entry and SEVONPEND = 0: – Any event: Refer to *Section 9.6.4: PWR exiting from low-power modes* If WFE was used for entry and SEVONPEND = 1: – Any Interrupt even when disabled in NVIC: refer to *Section 20.2: GIC Interrupts* or any event: refer to *Section 9.6.4: PWR exiting from low-power modes*
Wakeup latency	None

9.6.6 PWR CStop mode

The CStop mode applies to the MPU and/or MCU sub-systems. In CStop mode the MPU/MCU clock and MPU and/or MCU sub-systems peripherals clocks are stopped.

In CStop mode, MPU and/or MCU sub-systems peripherals having a kernel clock request are still able to request their kernel clock, when selecting a clock in the RCC which is able to run in CStop mode.

PWR entering to CStop mode

The CStop mode is entered according to *Section 9.6.3: PWR entering to low-power modes*,

- For MPU when both STPREQ_P[1:0] register bits in the RCC are set and the PDDS register bit in *PWR MPU control register (PWR_MPUCR)* is cleared.
  or
  When both STPREQ_P[1:0] register bits in the RCC are set and the CSTBYDIS register bit is set and the PDDS register bit in *PWR MPU control register (PWR_MPUCR)* is set.
- For MCU when the SLEEPDEEP bit in the Cortex®-M4 System Control register is set.

Refer to *Table 39* for MPU and *Table 40* for MCU, for details on how to enter to CStop mode.

PWR exiting from CStop mode

The CStop mode is exited according to *Section 9.6.4: PWR exiting from low-power modes*.

Refer to *Table 39* for MPU and *Table 40* for MCU, for more details on how to exit from CStop mode.

Table 39. MPU CStop

CStop mode	Description
Mode entry	WFI (Wait for Interrupt) – STPREQ_P = ‘b11 and MPU PDDS = 0 or STPREQ_P = ‘b11 and MPU CSTBYDIS = 1 – MPU GIC interrupts and events cleared. – All MPU EXTI Wakeup sources are cleared.
Mode exit	EXTI Interrupt enabled in GIC: Refer to *Section 20.2: GIC Interrupts*, for the peripherals which are not stopped.
Wakeup latency	EXTI and RCC wakeup synchronization (see *Section 10: Reset and clock control (RCC)*).

Table 40. MCU CStop

CStop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 1 (Refer to the Cortex®-M4 System Control register.) – MCU NVIC interrupts and events cleared. – All MCU EXTI Wakeup sources are cleared.
	On return from ISR while: – SLEEPDEEP = 1 and – SLEEPONEXIT = 1 (Refer to the Cortex®-M4 System Control register.) – MCU NVIC interrupts and events cleared. – All MCU EXTI Wakeup sources are cleared.
Mode exit(1)	If WFI or return from ISR was used for entry: – EXTI Interrupt enabled in NVIC: Refer to *Section 20.2: GIC Interrupts*, for peripheral which are not stopped. If WFE was used for entry and SEVONPEND = 0: – EXTI event: Refer to *Section 9.6.4: PWR exiting from low-power modes*, for peripheral which are not stopped. If WFE was used for entry and SEVONPEND = 1: – EXTI Interrupt even when disabled in NVIC: refer to *Section 20.2: GIC Interrupts* or EXTI event: refer to *Section 9.6.4: PWR exiting from low-power modes*, for peripheral which are not stopped.
Wakeup latency	EXTI and RCC wakeup synchronization (see *Section 10: Reset and clock control (RCC)*).

1. The IWDG1 and IWDG2 configured to generate a system debug wakeup event wake up the MPU and MCU, and trigger the debug CTI interface.

9.6.7 PWR CStandby mode

The CStandby mode applies only to the MPU sub-system. In CStandby mode the MPU clock and MPU sub-system peripherals clocks are stopped.

In CStandby mode, MPU sub-system peripherals having a kernel clock request are still able to request their kernel clock, when selecting a clock in the RCC which is able to run in CStop mode.

The MPU exits CStandby mode with a MPU reset.

PWR entering to CStandby mode

The CStandby mode is entered according to *Section 9.6.3: PWR entering to low-power modes*, when both MPU STPREQ_P[1:0] register bits in the RCC are set and the CSTBYDIS register bit is clear and the PDDS register bit in *PWR MPU control register (PWR_MPUCR)* is set.

Refer to *Table 41* for MPU, for details on how to enter to CStandby mode.

PWR exiting from CStandby mode

The CStandby mode is exited according to *Section 9.6.10: PWR Standby mode*.

Refer to *Table 41* for MPU, for more details on how to exit from CStandby mode.

Table 41. MPU CStandby

CStandby mode	Description
Mode entry	WFI (Wait for Interrupt) – STPREQ_P[1:0] = b’11 and MPU CSTBYDIS = 0 & PDDS = 1 – MPU GIC interrupts and events cleared. – All MPU EXTI wakeup sources are cleared.
Mode exit	EXTI Interrupt enabled in GIC: refer to *Section 20.2: GIC Interrupts*, for peripheral which are not stopped.
Wakeup latency	EXTI and RCC wakeup synchronization (see *Section 10: Reset and clock control (RCC)*).

9.6.8 PWR Stop mode

The system enters to Stop mode only when both the MPU and MCU sub-systems are in CStop mode or CStandby mode and the EXTI wakeup sources are inactive, and at least one of the MPU or MCU PDDS bit in *PWR MPU control register (PWR_MPUCR)* or *PWR MCU control register (PWR_MCUCR)* request Stop. In Stop mode, the bus matrix clocks are stalled. The oscillators can be stopped, for detailed oscillators status in Stop mode, see *Section 10.4.2: Oscillators description* and *Section 10.4.7: Clock generation in Stop and Standby modes*. For more information on the other clocks refer to RCC.

In Stop mode, the peripherals using the LSI or the LSE clock and peripherals having a kernel clock request selecting a clock source in the RCC which is able to run in Stop mode, are still able to operate.

In system Stop mode, the following features can be selected to be kept active by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by an Application reset. See *Section 46: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the *RCC reset duration and LSI control register (RCC_RDLSICR)*.
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the *RCC backup domain control register (RCC_BDCR)*.
- Peripherals capable of running on the LSI or LSE clock.
- Peripherals having a kernel clock request.
- Internal RC oscillators (HSI, CSI) and HSE oscillator: this is configured by the corresponding xxxKERON bit in the *RCC oscillator clock enable set register (RCC_OCENSETR)*
- The ADC or DAC can also consume power during the Stop mode, unless they are disabled before entering to it. To disable them, the ADON bit in the ADC_CR2 register and the ENx bit in the DAC_CR register must both be written to 0.

PWR entering to Stop mode

The Stop mode is entered according to *Section 9.6.3: PWR entering to low-power modes*, when at least one PDDS bit in *PWR MPU control register (PWR_MPUCR)* or *PWR MCU control register (PWR_MCUCR)* requests Stop.

Refer to *Table 42* for details on how to enter to Stop mode.

If an access to a bus matrix is ongoing, the Stop mode entry is delayed until the bus matrix access is finished.

PWR exiting from Stop mode

The Stop mode is exited according to *Section 9.6.4: PWR exiting from low-power modes*.

Refer to *Table 42* for more details on how to exit from Stop mode.

When exiting from Stop mode the RCC enables HSI for the MCU, and restores the MPU and AXI bus matrix clocks. Refer to *Section 10: Reset and clock control (RCC)* Leaving Stop mode.

The STOPF status flag in *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)* registers indicate that the system has exit from Stop mode. The MPU and MCU have their own set of flag.

Table 42. Stop

Stop mode	Description
Mode entry	– When the MPU and MCU are in CStop mode or CStandby mode, or the MCU is in HOLD_BOOT, and there is no active EXTI wakeup source. – At least one PDDS bit selects Stop. – LPDS selects Stop

Table 42. Stop (continued)

Stop mode	Description
Mode exit	– On an EXTI wakeup.
Wakeup latency	Clocks restored, see RCC for more information. + EXTI and RCC wakeup synchronization and PWRLP_TEMPO (see *Section 10: Reset and clock control (RCC)*).

PWR I/O states in Stop mode

In Stop mode, all I/O pins keep their configuration.

9.6.9 PWR LP-Stop and LPLV-Stop modes

The LP-Stop mode is handled in the same way as Stop mode, when LP-Stop mode is selected by the LPDS register bit in *PWR control register 1 (PWR_CR1)*. See *Section 9.6.8: PWR Stop mode*.

Refer to *Table 43* for more details on how to enter to and exit from LP-Stop mode.

Table 43. LP-Stop

Stop mode	Description
Mode entry	– When the MPU and MCU are in CStop mode or CStandby mode, or the MCU is in HOLD_BOOT, and there is no active EXTI wakeup source. – At least one PDDS bit selects Stop. – LPDS selects LP-Stop
Mode exit	– On an EXTI wakeup.
Wakeup latency	Clocks restored, see RCC for more information. + LP-Stop timeout (external regulator to switch to main mode) + EXTI and RCC wakeup synchronization and PWRLP_TEMPO (see *Section 10: Reset and clock control (RCC)*).

The LVDS register bit in *PWR control register 1 (PWR_CR1)* allows to select whether the V_{DDCORE} supply level during LP-Stop mode is kept at the Run mode level, or is lowered (LP-Stop with low voltage mode).

In LPLV-Stop mode, the peripherals able to wake up from Stop mode (such as UART, I2C, SPI) except LPTIM (using LSE or LSI), RTC, Tamper and wakeup pins, shall all be disabled, and the DDR retention shall be enabled by DDRRETEN in *PWR control register 3 (PWR_CR3)*.

Refer to *Table 44* for more details on how to enter to and exit from LPLV-Stop mode.

Table 44. LPLV-Stop

Stop mode	Description
Mode entry	– When the MPU and MCU are in CStop mode or CStandby mode, or the MCU is in HOLD_BOOT, and there is no active EXTI wakeup source. – At least one PDDS bit selects Stop – LPDS selects low power stop – LVDS selects LPLV-Stop
Mode exit	– WKUP pins, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, Application reset (NRST, IWDG, ...).
Wakeup latency	Clocks restored, see RCC for more information. + LPLV-Stop timeout (external regulator to restore Run mode operating supply level) + EXTI and RCC wakeup synchronization.

9.6.10 PWR Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the MPU and MCU sub-systems CStop mode and CStandby mode, as for the Stop mode, but with the V_{DDCORE} supply regulator powered off.

The system enters to Standby mode only when the MPU is in CStandby or CStop and the MCU is in CStop mode and both PDDS bits in *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)* request Standby. When the system enters to Standby mode the PWR_ON and PWR_LP pins are set low. It is expected that the External Voltage regulator providing the V_{DDCORE} supply is disabled, the complete V_{DDCORE} domain is consequently powered off. The PLLs, HSI, CSI oscillator and the HSE oscillator are also switched off, SRAM and register contents are lost except for the V_{DD}, VSW, backup domain, and retention domain (i.e. RTC registers, RTC backup register, LSE, backup RAM and Retention RAM), and Standby circuitry (see *Section 9.3.2: PWR supply system startup sequence*).

In system Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): the IWDG is started by writing to its Key register or by hardware option. Once started it cannot be stopped except by an Application reset. See *Section 46: Independent watchdog (IWDG)*.
- Real-time clock (RTC): this is configured by the RTCEN bit in the backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): this is configured by the LSION bit in the Control/status register (RCC_CSR).
- External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the backup domain control register (RCC_BDCR)

PWR entering to Standby mode

The Standby mode is entered according to *Section 9.6.3: PWR entering to low-power modes*, when all PDDS bits in *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)* request Standby.

Refer to *Table 45* for more details on how to enter to Standby mode.

PWR exiting from Standby mode

The Standby mode is exited according to *Section 9.6.4: PWR exiting from low-power modes*.

Refer to *Table 45* for more details on how to exit from Standby mode.

The system exits from Standby mode when an application reset (NRST, IWDG, ...), a WKUP pin event, a RTC alarm, a tamper event, or a time stamp event is detected. All registers are reset after wakeup from Standby except for the power control and status registers *PWR control register 2 (PWR_CR2)*, *PWR control register 3 (PWR_CR3)*, the SBF bit in *PWR MPU control register (PWR_MPUCR)* and *PWR MCU control register (PWR_MCUCR)*, *PWR wakeup flag register (PWR_WKUPFR)*, and *PWR MPU wakeup enable register (PWR_MPUWKUPENR)*.

After waking up from Standby mode, program execution restarts in the same way as after a Power On reset (boot option sampling, boot vector reset fetched, etc.). The MPU starts up booting and the MCU is kept in HOLD. From this point the MPU BOOTROM manages the startup (see *Processor hold from Standby* in *Section 10: Reset and clock control (RCC)*).

The exit from Standby/CStandby status flags can be retrieved thanks to the STDBYRSTF and CSTDBYRSTF bits in *RCC MPU reset status clear register (RCC_MP_RSTSCLRR)*. Those bits are set by the BOOTROM code from the SBF/SBFMPU bits in *PWR MPU control register (PWR_MPUCR)*. The SBF/SBFMPU bits are dedicated to the BOOTROM usage and should not be used elsewhere.

Table 45. Standby

Standby mode	Description
Mode entry	– The MPU sub-system is in CStandby or CStop and MCU sub-system is in CStop mode or HOLD_BOOT, and there is no active EXTI wakeup source. – All PDDS bits select Standby. – All WKUPF bits in Power Control/Status register (PWR_WKUPFR) are cleared. – All RTC wakeup events are cleared.
Mode exit	– WKUP pins, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, application reset (such as NRST, IWDG[(1)]).
Wakeup latency	System reset phase (see *Section 10: Reset and clock control (RCC)*).

1. The IWDG1 and IWDG2 when configured to generate an Application reset and when configured to generate a system debug wakeup event, reset in both cases the system.

PWR I/O states in Standby mode

In Standby mode, all I/O pins are high impedance no pull except for:

- NRST reset pin, PWR_ON pin, and PWR_LP pin (still available)
- RTC_AF1 pin if configured for tamper, time stamp, RTC Alarm out, or RTC clock calibration out
- WKUP pins, if enabled, the WKUP pin pulls can be configured by WKUPPUPD register bits in *PWR wakeup control register (PWR_WKUPCR)*.

9.6.11 PWR debug Stop

When the system enters low-power mode Stop, operation depend on the DBG_STOP configuration in the DBG module.

When the Stop mode debug is enabled, when entering Stop mode the PWR does not signal low-power mode on the PWR_ON and PWR_LP signals. This keeps the V_{DDCORE} domain to be supplied from the regulator in main mode.

When exiting from Stop mode the system MPU and logic is enabled as from normal Stop mode, the STOPF flag is set and when the MPU has entered CStandby mode the SBFMPU flag is set as well.

9.6.12 PWR debug Standby

When the system enters low-power mode Standby, operation depend on the DBG_Standby configuration in the DBG module.

When the Standby mode debug is enabled, when entering Standby mode the PWR keeps the pwr_on active, allowing the V_{DDCORE} domain to remain supplied. In Standby mode the VDDCORE domain is not reset, allowing the CPU and bus clocks to be active.

When exiting from Standby mode a reset is generated, which resets the V_{DDCORE} logic excluding the debug logic. Waking up from debug Standby mode the system MPU and MCU and logic is reset as from normal Standby mode and the SBF flag are set. When the MPU has entered CStandby mode the SBFMPU flag is set as well.

9.7 PWR WKUP pins

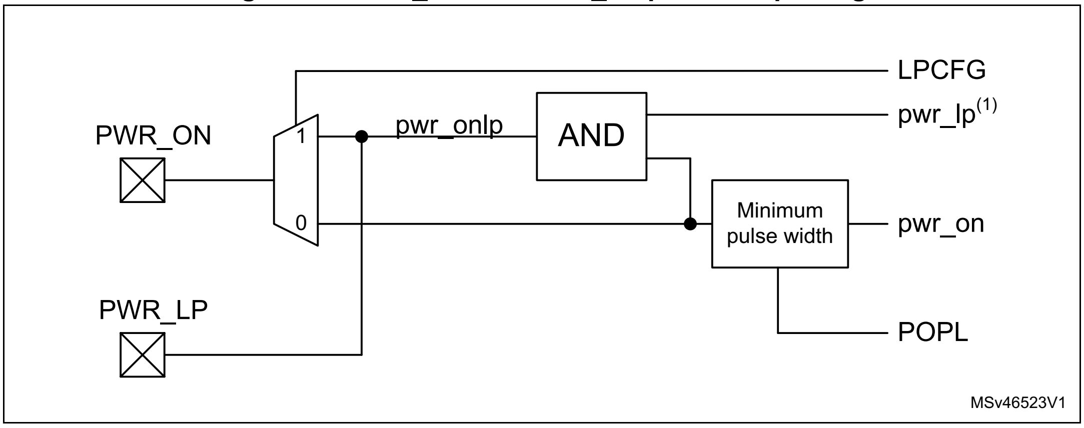
The WKUP polarity shall only be changed in WKUPP when the wakeup interrupt is disabled, in WKUPEN.

To avoid falls interrupts, the WKUPF shall be cleared with WKUPC, before enabling the wakeup interrupt in WKUPEN.

9.8 PWR_ON and PWR_LP pins

The PWR_ON pin may signal pwr_on or pwr_onlp depending the selection in LPCFG register bit in *PWR control register 1 (PWR_CR1)*. The PWR_LP pin signals pwr_onlp.

Figure 45. PWR_ON and PWR_LP pins multiplexing



1. See LPDS bit in *PWR control register 1 (PWR_CR1)*.

PWR_ON pin

When LPCFG select pwr_on signaling, the PWR_ON pad is only low in Standby mode.

When LPCFG select pwr_onlp signaling on PWR_ON, the definition (Standby, LP-Stop or LPLV-Stop) of the signal being low shall be communicated to the external power management prior entering low-power mode. The PWR_ON pad is low in Standby, LP-Stop and LPLV-Stop modes.

9.9 PWR interrupts

The PWR provides the following interrupts:

- Wakeup interrupt to MPU.
  - Enabled by WKUPEN[6:1] register bit in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)*.
  - The interrupt status flag WKUPF[6:1] are available from *PWR wakeup flag register (PWR_WKUPFR)*.
- Wakeup interrupt to MCU.
  - Enabled by WKUPEN[6:1] register bit in *PWR MCU wakeup enable register (PWR_MCUWKUPENR)*.
  - The same interrupt status flag WKUPF[6:1] as for the MPU are available from *PWR wakeup flag register (PWR_WKUPFR)*.

Other status flag associated with Stop and Standby modes and monitoring functions are also available, see *Table 46*:

Table 46. PWR status flags

Status Flag	Description	Interrupt	Clear through
STOPF	Exit from Stop mode	Wakeup from Stop mode events	CSSF
SBF[1]	Exit from Standby mode	Wakeup from Standby events.	CSSF

Table 46. PWR status flags (continued)

Status Flag	Description	Interrupt	Clear through
SBFMPU[(1)]	MPU exit CStandby mode	Wakeup from CStandby, events.	CSSF
WKUPF[6:1]	Wakeup flags	MPU and MCU wakeup from Standby and Stop events.	WKUPC[6:1]
PVDO	Programmable voltage detector	interrupt via EXTI	n.a.
AVDO	Analog voltage detector	interrupt via EXTI	n.a.
TEMPL, TEMPH	Temperature monitoring	interrupt via EXTI	n.a.
BRRDY	Backup regulator ready	no associated interrupt	n.a.
RRRDY	Retention regulator ready	no associated interrupt	n.a.
USB33RDY	USB FS 3V3 supply ready	no associated interrupt	n.a.
REG18RDY	Regulator 1V8 supply ready	no associated interrupt	n.a.
REG11RDY	Regulator 1V1 supply ready	no associated interrupt	n.a.

1. SBF and SBFMPU are dedicated to the BOOTROM usage and should not be used elsewhere. The STDBYRSTF and CSTDBYRSTF bits in *RCC MPU reset status clear register (RCC_MP_RSTSCLRR)* should be used by application code as part of reset source identification process. See *Section 10.3.13: Reset source identification*.

9.10 PWR TrustZone security

The PWR is able to secure registers from being modified by non-secure accesses. The TrustZone security is activated by the RCC TZEN register bit in *Section 10.7.2: RCC TrustZone control register (RCC_TZCR)*. At PWR level the security consist in preventing a non-secure write access to:

- Change the settings of the VBAT and TEMP Monitor, PVD and AVD.
- Change the low-power Deepsleep and RAM low-power settings.
- Change the backup domain write protection.
- Change the Backup regulator, Retention regulator, 1V8 regulator, 1V1 regulator and USB 3.3V voltage level detector settings.
- Change the backup battery charging settings.
- Change the MPU power control register settings.
- Change the Standby wakeup settings and flags.

The PWR TrustZone security enables write access security for all features simultaneously.

Table 47. Register security overview

Register name	Access type	Security[(1)]
PWR_CR1	RW	Write security
PWR_CSR1	R	Non secure
PWR_CR2	RW	Write secure
PWR_CR3	RW	Write secure

Table 47. Register security overview (continued)

Register name	Access type	Security[1]
PWR_MPUCR	RW	Write secure
PWR_MCUCR	RW	Non secure
PWR_WKUPCR	RW	Write secure[2]
PWR_WKUPFR	R	Non secure
PWR_MPUWKUPENR	RW	Write secure
PWR_MCUWKUPENR	RW	Non secure
PWR_VER	R	Non secure
PWR_ID	R	Non secure
PWR_SID	R	Non secure

1. Security is enabled with the RCC TZEN register bit.
2. Security is additionally channel wise enabled with WKUPEN bits in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* register.

9.11 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless explicitly specified differently.

9.11.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

Reset on any system reset.

This register provides write access security when enabled by TZEN register bit in *Section 10: Reset and clock control (RCC)*. When security is enabled a non-secure write access generates a bus error. Secure and non-secure read accesses are granted and return the register value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS[1:0]		AVDEN
													rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP	PLS[2:0]			PVDEN	Res.	LVDS	LPCFG	LPDS
							rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **ALS[1:0]:** Analog Voltage Detector level selection.

These bits select the voltage threshold detected by the AVD.
00: 1.7V
01: 2.1V
10: 2.5V
11: 2.8V

Bit 16 **AVDEN:** Peripheral Voltage Monitor on V_{DDA} enable.

0: Peripheral Voltage Monitor on V_{DDA} disabled
1: Peripheral Voltage Monitor on V_{DDA} enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DBP:** Disable backup domain write protection.

In reset state, the RCC_BDCR, PWR_CR2, RTC, and backup registers are protected against parasitic write access. This bit must be set to enable write access to these.
0: Write access to RTC and backup domain registers disabled.
1: Write access to RTC and backup domain registers enabled.

Bits 7:5 **PLS[2:0]:** Programmable Voltage Detector level selection.

These bits are read only when the SYSCFG register bit PVDL is set (when PVDL is set, there is no bus errors generated when writing this register).
These bits select the voltage threshold detected by the PVD.
000: 1.95 V
001: 2.1 V
010: 2.25 V
011: 2.4 V
100: 2.55 V
101: 2.7 V
110: 2.85 V
111: External voltage level on PVD_IN (compared to internal VREFINT)

Bit 4 **PVDEN:** Programmable Voltage detector enable.

This bit is read only when the SYSCFG register bit PVDL is set (when PVDL is set, there is no bus errors generated when writing this register).
0: Programmable Voltage detector disabled.
1: Programmable Voltage detector enabled.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **LVDS:** Low Voltage Deepsleep LPLV-Stop mode selection.

This bit has only effect when the low power stop mode is selected in LPDS and changes the V_{DDCORE} domain supply reset level.

0: LP-Stop mode V_{DDCORE} domain supply reset level at same level as Run mode. The V_{DDCORE} domain supply level in LP-Stop mode shall be kept at same level as Run mode.

1: LPLV-Stop mode V_{DDCORE} domain supply reset level at lower level than Run mode. Allows to lower V_{DDCORE} domain supply level in LPLV-Stop mode (see the datasheet for voltage levels).

Bit 1 **LPCFG:** PWR_ON pin configuration.

0: PWR_ON pin signals Standby mode (PWR_ON =1 in Run, Stop, LP-Stop, LPLV-Stop, and 0 in Standby).

1: PWR_ON pin signals Standby, LP-Stop and LPLV-Stop modes (PWR_ON = 1 in Run, Stop, and 0 in LP-Stop, LPLV-Stop and Standby).

Bit 0 **LPDS:** Low Power Deepsleep Stop mode selection.

0: Stop mode selected, External regulator kept in Main power mode (pwr_lp = 1).

1: low-power stop mode selected External regulator may enter Low-power mode (pwr_lp = 0). Further low power mode selection is provided by LVDS.

9.11.2 PWR control status register 1 (PWR_CSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Reset on any system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVDO
															r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.
											r				

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AVDO**: Analog Voltage detector Output on V_{DDA}.

This bit is set and cleared by hardware. It is valid only if AVD on V_{DDA} is enabled by the AVDEN bit.

0: V_{DDA} is equal or higher than the AVD threshold selected with the ALS[2:0] bits.

1: V_{DDA} is lower than the AVD threshold selected with the ALS[2:0] bits

Note: The AVD is disabled in Standby mode and after system reset. For this reason, this bit is equal to 0 after Standby and reset.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **PVDO**: Programmable Voltage Detect Output

This bit is set and cleared by hardware. It is valid only if PVD is enabled by the PVDEN bit.

0: V_{DD} or voltage level on PVD_IN is equal or higher than the PVD threshold selected with the PLS[2:0] bits.

1: V_{DD} or voltage level on PVD_IN is lower than the PVD threshold selected with the PLS[2:0] bits.

Note: The PVD is disabled in Standby mode and after a system reset. For this reason, this bit is equal to 0 after Standby and system reset.

Bits 3:0 Reserved, must be kept at reset value.

9.11.3 PWR control register 2 (PWR_CR2)

Address offset: 0x008

Reset value: 0x0000 0000

Not reset by wakeup from Standby mode, Application reset (NRST, IWDG, ...) and V_{DD} POR, but reset only by V_{SW} POR and VSWRST.

Access 6 wait states when writing this register.

After reset the register is write-protected and the DBP bit in the *PWR control register 1 (PWR_CR1)* has to be set before it can be written. When DBP is cleared, there is no bus errors generated when writing this register.

This register shall not be accessed when the RCC VSWRST register bit in *Section 10.7.86: RCC backup domain control register (RCC_BDCR)* resets the VSW domain.

This register provides Write access security when enabled by TZEN register bit in *Section 10.7.2: RCC TrustZone control register (RCC_TZCR)*. When security is enabled a non-secure write access generates a bus error. Secure and non-secure read accesses are granted and return the register value.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	Res.	Res.	Res.	Res.	RRRDY	BRRDY
								r	r					r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.	Res.	RREN	BREN
											rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TEMPH**: Monitored temperature level above high threshold

0: Temperature below high threshold level, or Monitor disabled.

1: Temperature equal or above high threshold level.

Bit 22 **TEMPL**: Monitored temperature level above low threshold

0: Temperature above low threshold level, or Monitor disabled.

1: Temperature equal or below low threshold level.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **RRRDY**: Retention Regulator ready

Set by hardware to indicate that the Retention Regulator is ready. Entering Standby or VBAT mode before the Retention Regulator is ready may cause loss of Retention RAM content.

0: Retention Regulator not ready.

1: Retention Regulator ready.

Bit 16 **BRRDY**: Backup Regulator ready

Set by hardware to indicate that the Backup Regulator is ready. Entering Standby or VBAT mode before the Backup Regulator is ready may cause loss of Backup RAM content.

0: Backup Regulator not ready.

1: Backup Regulator ready.

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **MONEN:** VBAT and temperature monitoring enable

When set, the V_{BAT} supply and temperature monitoring is enabled.

0: V_{BAT} and temperature monitoring disabled.

1: V_{BAT} and temperature monitoring enabled.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RREN:** Retention regulator enable

When set, the retention regulator (used to maintain retention RAM content in Standby and VBAT modes) is enabled.

If RREN is reset, the Retention Regulator is switched off. The retention RAM can still be used in Run mode and Stop mode but its content is lost in the Standby and VBAT modes. Once set, the application must wait that the retention regulator Ready flag (RRRDY) is set to indicate that the data written into the SRAM is maintained in the Standby and VBAT modes.

0: Retention regulator disabled

1: Retention regulator enabled

Bit 0 **BREN:** Backup regulator enable

When set, the backup regulator (used to maintain backup RAM content in Standby and VBAT modes) is enabled.

If BREN is reset, the backup regulator is switched off. The backup RAM can still be used in Run mode and Stop mode but its content is lost in the Standby and VBAT modes. Once set, the application must wait that the Backup Regulator Ready flag (BRRDY) is set to indicate that the data written into the SRAM is maintained in the Standby and VBAT modes.

0: Backup regulator disabled

1: Backup regulator enabled

9.11.4 PWR control register 3 (PWR_CR3)

Address offset: 0x00C

Reset value: 0x5000 0000

Not reset by wakeup from Standby mode and Application reset (such as NRST, IWDG) but only reset by V_{DD} POR.

Access 6 wait states when writing this register.

This register provides Write access security when enabled by TZEN register bit in *Section 10: Reset and clock control (RCC)*. When security is enabled a non-secure write access generates a bus error. Secure and non-secure read accesses are granted and return the register value.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG11 RDY	REG11 EN	REG18 RDY	REG18 EN	Res.	USB33 RDY	Res.	USB33 DEN	Res.	Res.	POPL[4:0]					Res.
r	rw	r	rw		r		rw			rw	rw	rw	rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	DDRR ETEN	DDRS RDIS	DDRS REN	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw								

Bit 31 **REG11RDY**: 1V1 regulator supply ready
0: 1V1 supply not ready
1: 1V1 supply ready

Bit 30 **REG11EN**: 1V1 regulator enable
0: 1V1 regulator disabled
1: 1V1 regulator enabled (when in Standby mode the 1V1 regulator is disabled)

Bit 29 **REG18RDY**: 1V8 regulator supply ready
0: 1V8 supply not ready
1: 1V8 supply ready

Bit 28 **REG18EN**: 1V8 regulator enable
0: 1V8 regulator disabled
1: 1V8 regulator enabled, when also pin BYPASS_REG1V8 is low (when in Standby mode the 1V8 regulator is disabled).

Bit 27 Reserved, must be kept at reset value.

Bit 26 **USB33RDY**: USB 3.3V supply ready
0: USB 3.3V supply not ready
1: USB 3.3V supply ready

Bit 25 Reserved, must be kept at reset value.

Bit 24 **USB33DEN**: USB 3.3V voltage level detector enable
0: USB 3.3V voltage level detector disabled
1: USB 3.3V voltage level detector enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:17 **POPL[4:0]**: pwr_on pulse low configuration
These bits are set and cleared by software. They define the minimum guaranteed duration of the pwr_on low pulse in Standby mode (there is no impact on the LP-Stop and LPLV-Stop modes).
The LSI oscillator is automatically enabled when needed by the POPL pulse low configuration.
00000: No guaranteed minimum low time
00001: ~ 1 ms guaranteed minimum low time (1 x 32 LSI cycles)
00010: ~ 2 ms guaranteed minimum low time (2 x 32 LSI cycles)
...
11111: ~ 31 ms guaranteed minimum low time (31 x 32 LSI cycles)

Bits 16:13 Reserved, must be kept at reset value.

Bit 12 **DDRRETEN:** DDR retention enable
Set by Hardware on NRST pad reset, when the reset delay control is enabled in the RCC_MP_APRSTCR.RDCTLEN register bit, or when written ‘b1 by software, the DDR PHY puts its pads in retention.
To disable retention software shall write this bit to ‘b0.
0: DDR self-refresh retention disabled.
1: DDR self-refresh retention enabled.

Bit 11 **DDRSRDIS:** DDR self-refresh retention after standby disable
When written ‘b1, the DDR PHY pads retention, enabled due to DDRSREN, is disabled.
0: No action
1: When DDRSREN is ‘b1: Disable DDR self-refresh retention after standby mode. This bit is reset by hardware once DDR self-refresh retention has been disabled.
When DRRSREN is ‘b0 no action.

Bit 10 **DDRSREN:** DDR self-refresh in standby mode enable
When set, the DDR PHY puts its pads in retention when entering Standby mode.
0: DDR self-refresh retention when entering standby mode disabled.
1: DDR self-refresh retention when entering standby mode enabled (DDR self-refresh retention after Standby shall be disabled by DDRSRDIS).

Bit 9 **VBRS**: VBAT charging resistor selection
0: Charges VBAT through a 5 kOhm resistor.
1: Charges VBAT through a 1.5 kOhm resistor.

Bit 8 **VBE**: VBAT charging enable
0: VBAT battery charging disabled.
1: VBAT battery charging enabled.

Bits 7:0 Reserved, must be kept at reset value.

9.11.5 PWR MPU control register (PWR_MPUCR)

Address offset: 0x010

Reset value: 0x0000 0000

See individual bits for reset condition.

Access 6 wait states when writing this register.

This register provides Write access security when enabled by TZEN register bit in *Section 10: Reset and clock control (RCC)*. When security is enabled a non-secure write access generates a bus error. Secure and non-secure read accesses are granted and return the register value.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STANDBYWFIL2	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	SBFMPU	SBF	STOPF	Res.	CSTBYDIS	Res.	Res.	PDDS
r						rw		r	r	r		rw			rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **STANDBYWFIL2:** MPU system idle indication
This bit is set and reset by hardware based on the MPU operation mode.
0: MPU system CRun or CSleep
1: MPU system CStop or CStandby

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **CSSF:** Clear MPU Standby, Stop flags.(Always read as 0)
This bit is reset on any system reset.
0: No effect
1: When written, clears the MPU flags (STOPF, SBF, and SBFMPU). The register bit is cleared to 0 by hardware.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **SBFMPU:** MPU Standby flag
This bit is set by hardware and cleared only by a V_{DD} POR reset or by setting the MPU CSSF bit (not reset when exit from Standby mode). This bit is cleared by the MPU Boot ROM.
0: MPU has not been in CStandby mode.
1: MPU has been in CStandby mode.

Bit 6 **SBF:** System Standby flag
This bit is set by hardware and cleared only by a V_{DD} POR reset or by setting the MPU CSSF bit (not reset when exit from Standby mode). This bit is cleared by the MPU Boot ROM.
0: System has not been in Standby mode.
1: System has been in Standby mode, system contents has been lost.

Bit 5 **STOPF:** Stop flag
This bit is set by hardware and cleared only by any system reset or by setting the MPU CSSF bit. This bit is cleared by the MPU Boot ROM.
0: System has not been in Stop mode
1: System has been in Stop mode, clock system has been stopped.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **CSTBYDIS:** MPU CStandby mode disable
This bit is reset only by a V_{DD} POR reset (not reset when exit from Standby mode).
Allows to define the use of the MPU CStandby mode. Under some conditions, this bit is cleared by the MPU Boot ROM.
0: MPU CStandby mode enabled
1: MPU CStandby mode disabled

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **PDDS:** System Power Down Deepsleep selection
This bit is reset only by a V_{DD} POR reset (not reset when exit from Standby mode).
Allows MPU to define the Deepsleep mode for the system.
0: Keeps Stop mode when MPU enters to CStop.
1: Allows Standby mode when MPU enters to CStop. When CSTBYDIS = 0 allows MPU to enter CStandby mode.

9.11.6 PWR MCU control register (PWR_MCUCR)

Address offset: 0x014

Reset value: 0x0000 0000

See individual bits for reset condition.

Access 6 wait states when writing this register.

This register is always non-secure.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEEPSLEEP	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	PDDS
r						rw			r	r					rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DEEPSLEEP:** MCU system idle indication

This bit is set and reset by hardware based on the MCU operation mode.
0: MCU system CRun or CSleep
1: MCU system CStop

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **CSSF:** Clear MCU Standby, Stop flags (always read as 0)

This bit is reset on any system reset.
0: No effect
1: When written, clear the MCU flags (STOPF, and SBF). Register bit is cleared to 0 by hardware.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **SBF:** System Standby Flag

This bit is set by hardware and cleared only by a V_{DD} POR reset or by setting the MCU CSSF bit (not reset when exit from Standby mode).
0: System has not been in Standby mode.
1: System has been in Standby mode, system contents has been lost.

Bit 5 **STOPF:** Stop Flag

This bit is set by hardware and cleared only by any system reset or by setting the MCU CSSF bit.
0: System has not been in Stop mode.
1: System has been in Stop mode, clock system has been stopped.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **PDDS:** System Power Down Deepsleep selection.

This bit is reset only by a V_{DD} POR reset (not reset when exit from Standby mode).
Allows MCU to define the Deepsleep mode for the system.
0: Keeps Stop mode when MCU enters to CStop.
1: Allows Standby mode when MCU enters to CStop.

9.11.7 PWR wakeup control register (PWR_WKUPCR)

Address offset: 0x020

Reset value: 0x0000 0000

Not reset by wakeup from Standby mode, but by any application reset (such as NRST, IWDG).

Access 6 wait states when writing this register (when clearing a WKUPF, the AHB write access completes after the WKUPF has cleared).

This register provides Write access security when enabled by TZEN register bit in *Section 10: Reset and clock control (RCC)*. When security is enabled a non-secure write access on individual WKUPC[6:1], WKUPP[6:1] bits and WKUPPUPD[6:1] bit pairs are discarded when the corresponding WKUPEN[6:1] bit in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is set. No bus error is generated. Secure and non-secure read accesses are granted and return the register value.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WKUPPUPD6 [1:0]		WKUPPUPD5 [1:0]		WKUPPUPD4 [1:0]		WKUPPUPD3 [1:0]		WKUPPUPD2 [1:0]		WKUPPUPD1 [1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	WKUP P6	WKUP P5	WKUP P4	WKUP P3	WKUP P2	WKUP P1	Res.	Res.	WKUP C6	WKUP C5	WKUP C4	WKUP C3	WKUP C2	WKUP C1
		rw	rw	rw	rw	rw	rw			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WKUPPUPD[6:1][1:0]**: Wakeup pull configuration for WKUPn pin, n range [6:1]

These bits define the IO pad pull configuration used when WKUPENn = 1 (note that the associated GPIO port pull configuration shall be set to the same value or 00). The wakeup pin pull configuration is maintained in Standby mode.

When TZEN is enabled in the RCC and WKUPENn in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is set, bit pair n can only be written by a secure access, a non-secure write on this bit pair is discarded.

When TZEN is disabled in the RCC or WKUPENn in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is clear, bit pair n can be written by a secure and non-secure access.

00: No pulls
01: Pull-up
10: Pull-down
11: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **WKUPP[6:1]**: Wakeup Polarity bit for WKUPn pin, n range [6:1]

These bits define the polarity used for event detection on external wake-up WKUPn pin.

When TZEN is enabled in the RCC and WKUPENn in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is set, bit n can only be written by a secure access, a non-secure write on this bit is discarded.

When TZEN is disabled in the RCC or WKUPENn in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is clear, bit n can be written by a secure and non-secure access.

0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPC[6:1]:** Clear Wakeup Flag for WKUPn pin, n range [6:1]

When TZEN is enabled in the RCC and WKUPENn in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is set, bit n can only be written by a secure access, a non-secure write on this bit is discarded.

When TZEN is disabled in the RCC or WKUPENn in *PWR MPU wakeup enable register (PWR_MPUWKUPENR)* is clear, bit n can be written by a secure and non-secure access.

These bits are always read as 0.

0: No effect
1: Writing 1 clears the WKUPFn wakeup flag (the bit is cleared to 0 by hardware).

9.11.8 PWR wakeup flag register (PWR_WKUPFR)

Address offset: 0x024

Reset value: 0x0000 0000

Not reset by wakeup from Standby mode but by any Application reset (NRST, IWDG, ...)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF6	WKUPF5	WKUPF4	WKUPF3	WKUPF2	WKUPF1
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPF[6:1]:** Wakeup flag for WKUPn pin before enable, n range [6:1].
This bit is set by hardware and cleared only by a NRST Reset or by setting the WKUPCn bit in the *PWR wakeup control register (PWR_WKUPCR)*
0: No wakeup event occurred
1: A wakeup event was received from WKUPCn pin

9.11.9 PWR MPU wakeup enable register (PWR_MPUWKUPENR)

Address offset: 0x028

Reset value: 0x0000 0000

Not reset by wakeup from Standby mode but by any Application reset (NRST, IWDG, ...).

Access 6 wait states when writing this register.

This register provides Write access security when enabled by TZEN register bit in *Section 10: Reset and clock control (RCC)*. When security is enabled a non-secure write access is discarded and a bus error is generated. Secure and non-secure read accesses are granted and return the register value.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPEN[6:1]**: Enable Wakeup WKUPn pin and interrupt for MPU, n range [6:1]
Each bit is set and cleared by software.
When TZEN is enabled in the RCC, these bits can only be written by a secure access, a non-secure write is discarded.
When TZEN is disabled in the RCC, these bits can be written by a secure and non-secure access.
0: An event on WKUPn pin does not wake up the system from Standby and Stop mode, nor generates an interrupt to MPU.
1: A rising or falling edge on WKUPn pin wakes up the system from Standby and Stop mode and generates an interrupt to MPU.

9.11.10 PWR MCU wakeup enable register (PWR_MCUWKUPENR)

Address offset: 0x02C

Reset value: 0x0000 0000

Not reset by wakeup from Standby mode but by any Application reset (NRST, IWDG, ...)

Access 6 wait states when writing this register.

When a system reset occurs during the register write cycle the written data is not guaranteed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **WKUPEN[6:1]**: Enable Wakeup WKUPn pin and interrupt for MCU, n range [6:1]
Each bit is set and cleared by software.
0: An event on WKUPn pin does not wake up the system from Standby and Stop mode, nor generate an interrupt to MCU.
1: A rising or falling edge on WKUPn pin wakes up the system from Standby and Stop mode, and generate an interrupt to MCU.

9.11.11 PWR IP version register (PWR_VER)

Address offset: 0x3F4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision number

Bits 3:0 **MINREV[3:0]:** Minor revision number

9.11.12 PWR IP identification register (PWR_ID)

Address offset: 0x3F8

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]:** IP identification

9.11.13 PWR size ID register (PWR_SID)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

9.11.14 PWR register map

Table 48. Power control register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS[1:0]		AVDEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP	PLS[2:0]			PVDEN	Res.	LVDS	LPCFG	LPDS
	Reset value														0	0	0								0	0	0	0	0		0	0	0
0x004	**PWR_CSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVDO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.
	Reset value																0												0				
0x008	**PWR_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	Res.	Res.	Res.	Res.	RRRDY	BRRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	Res.	Res.	RREN	BREN
	Reset value									0	0					0	0												0			0	0
0x00C	**PWR_CR3**	REG11RDY	REG11EN	REG18RDY	REG18EN	Res.	USB33RDY	Res.	USB33DEN	Res.	Res.	POPL[4:0]					Res.	Res.	Res.	Res.	DDRRETEN	DDRSRDIS	DDRSREN	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	1	0	1		0		0			0	0	0	0	0					0	0	0	0	0								

Table 48. Power control register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	**PWR_MPUCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STANDBYWFIL2	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	SBFMPU	SBF	STOPF	Res.	CSTBYDIS	Res.	Res.	PDDS
	Reset value																	0						0		0	0	0		0			0
0x014	**PWR_MCUCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEEPSLEEP	Res.	Res.	Res.	Res.	Res.	CSSF	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	PDDS
	Reset value																	0						0			0	0					0
0x018-0x01C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x020	**PWR_WKUPCR**	Res.	Res.	Res.	Res.	WKUPPUPD6 [1:0]		WKUPPUPD5 [1:0]		WKUPPUPD4 [1:0]		WKUPPUPD3 [1:0]		WKUPPUPD2 [1:0]		WKUPPUPD1 [1:0]		Res.	Res.	WKUPP6	WKUPP5	WKUPP4	WKUPP3	WKUPP2	WKUPP1	Res.	Res.	WKUPC6	WKUPC5	WKUPC4	WKUPC3	WKUPC2	WKUPC1
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x024	**PWR_WKUPFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF6	WKUPF5	WKUPF4	WKUPF3	WKUPF2	WKUPF1
	Reset value																											0	0	0	0	0	0
0x028	**PWR_ MPUWKUPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
	Reset value																											0	0	0	0	0	0
0x02C	**PWR_ MCUWKUPENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPEN6	WKUPEN5	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
	Reset value																											0	0	0	0	0	0
0x030-0x3F0	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x3F4	**PWR_VER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0x3F8	**PWR_ID**	IPID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3FC	**PWR_SID**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

10 Reset and clock control (RCC)

The RCC block is responsible for the management of the clock and reset generation for the complete circuit.

This section often refers to operating modes defined in *Section 9.5.1: PWR operating modes*.

10.1 RCC main features

- AHB-Lite bus interface
- TrustZone aware
- Register protection function
- Reset part:
  - Generation of local and system reset
  - Bidirectional pad reset (NRST) allowing the reset of external devices, or the reset of the circuit
- Clock generation part:
  - Generation and distribution of clocks for the complete system
  - 4 separate PLLs:
    - Integer or fractional mode
    - Spread spectrum function to reduce the amount of EMI peaks
    - Possibility to change the PLL fractional ratios on-the-fly
  - Smart clock gating for reduction of power dissipation
  - 2 external oscillators:
    - HSE, supporting a wide range of crystals: 8 to 48 MHz
    - LSE, for the 32.768 kHz crystal
  - 3 Internal oscillators: HSI, CSI and LSI
  - Buffered clock outputs for external devices
  - Generation of 2 interrupts lines:
    - A dedicated interrupt line for clock security management
    - A general interrupt line for other events
- 2 Independent interrupt interfaces
- 2 Independent events to wake up processors

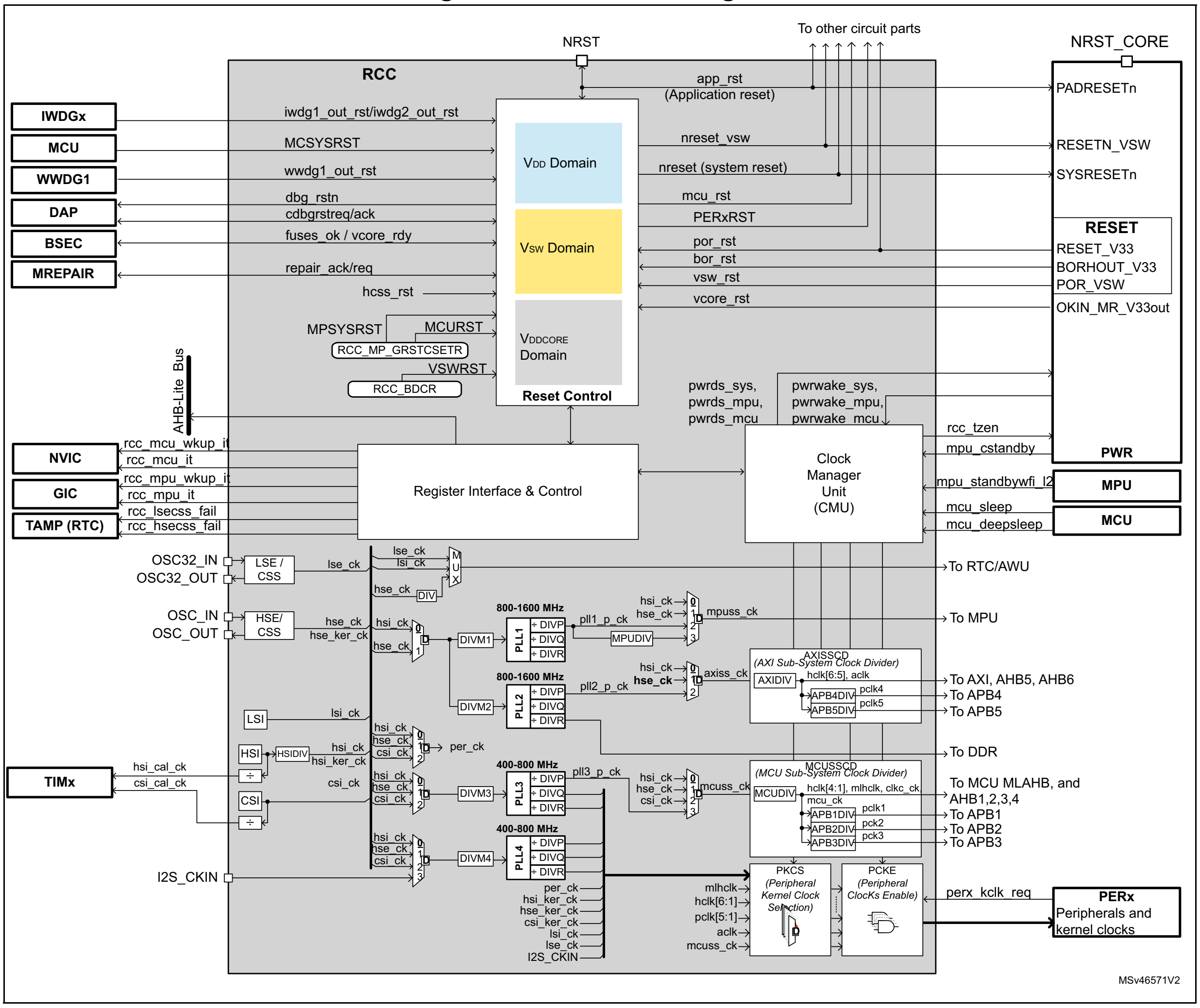
10.2 RCC block diagram

Figure 46 gives a simplified view of the RCC, and the key signals exchanged with the other blocks. Three voltage domains are present in the RCC: the V_{DD} domain (blue), the V_{SW} domain (yellow) and the V_{DDCORE} domain (gray).

For the reset part, analog comparators located into the PWR block provide all reset signals dependent of the power supplies: **por_rst**, **bor_rst**, **vcore_rst** and **vsw_rst**. Two reset sources are coming from pads: NRST and NRST_CORE. The RCC also receives resets from various other sources: watchdogs, MCU, registers... All these resets are handled by the RCC in order to reset properly all the parts of the system.

For the clock generation part, the RCC handles the generation and the gating for all the clocks according to register settings and processors modes. Note that the RCC is clocked with hclk4 clock.

Figure 46. RCC block diagram



The RCC exchanges also some signals with the PWR in order to control the entry and the exit of the system in Stop and Standby modes.

10.2.1 RCC pin overview

Table 49 lists the RCC signals connected to pads or balls.

Table 49. Pin overview

Pin Name	I/O	Description
NRST	I/O	External (application) reset, can be used to provide reset to external devices
OSC32_IN	I	32.768 kHz oscillator input or external clock input
OSC32_OUT	O	32.768 kHz oscillator output
OSC_IN	I	HSE oscillator input or external clock input
OSC_OUT	O	HSE oscillator output
MCO1	O	Output clock 1 for external devices
MCO2	O	Output clock 2 for external devices
I2S_CKIN	I	External kernel clock input for digital audio interfaces: SPI/I2S, SAI, and DFSDM
ETH_TX_CLK	I	External clock TX provided by the Ethernet PHY for MII
ETH_RX_CLK/ ETH_REF_CLK	I	External clock RX provided by the Ethernet PHY for MII, RMII, GMII and RGMII
ETH_CLK	O	Clock output for external Ethernet PHY
ETH_CLK125	I	External clock RX provided by the Ethernet PHY for GMII and RGMII
GTX_CLK	O	Clock output for external Ethernet PHY

10.3 RCC functional description - reset part

There are several sources able to generate a reset:

- Supply voltages (V_{DD} or V_{DDCORE} or V_{SW}) lower than the expected values,
- Watchdog timeout,
- MPU exit from CStandby,
- Product exit from Standby,
- External signals driving the NRST or NRST_CORE pads,
- Software commands.

The coverage (or scope) of the resets differs according to the source initiating the reset. However the following categories can be distinguished:

- The power-on/off resets,
- The application and system resets,
- The local resets.

10.3.1 The power-on/off resets

The PWR block provides some reset signals to the RCC:

- The **por_rst** signal, which is asserted when the V_{DD} supply is lower than the V_{POR} threshold.
- The **bor_rst** signal, which is asserted when the V_{DD} supply is lower than the V_{BOR} threshold.
- The **vcore_rst** signal, which is asserted when the V_{DDCORE} supply is lower than the V_{DDCORE} threshold, or when the pad NRST_CORE is asserted.
- The **vsw_rst** signal, which is asserted when the V_{SW} supply is lower than the expected threshold.

As shown in *Figure 47*, the signal provided by the PWR block, indicating that the V_{DDCORE} is valid (OKIN_MR_V33out), is used by the RCC in order to trigger the hardware system initialization sequence and the memory repair. Refer to *Section 3: Boot and security and OTP control (BSEC)* for details. When this initialization sequence is completed, then the **vcore_rst** is released allowing the processors to start.

Refer to *Table 50: Reset coverage summary* for details.

Refer to *Section 9: Power control (PWR)* for details on the generation of power-on/off resets.

10.3.2 The application and system resets

An application reset (**app_rst**) is generated from one of the following sources:

- A reset from NRST pad,
- A reset from **por_rst** signal (generally called power-on reset),
- A reset from **bor_rst** signal (generally called brownout),
- A reset from the Independent Watchdogs 1 (**iwdg1_out_rst**),
- A reset from the Independent Watchdogs 2 (**iwdg2_out_rst**),
- A software reset from the Cortex-M4 (MCU), via the AIRCR register (MCSYSRST) if the option byte OTP_MCU_SYSRST_EN allows it,
- A software reset from the RCC, when the Cortex-A7 (MPU) sets the bit MPSYSRST to '1' via *RCC global reset control set register (RCC_MP_GRSTCSETR)*,
- A failure on HSE, when the clock security system feature is activated (**hcss_rst**).

A System reset (**nreset**) is generated from one of the following sources:

- A reset from **app_rst** signal (application reset),
- A reset from **vcore_rst** signal.

Note: When the system is in Standby, the V_{DDCORE} is switched off, but V_{DD} is still present. So when the system exits from Standby, the vcore_rst signal is activated, generating a nreset reset.

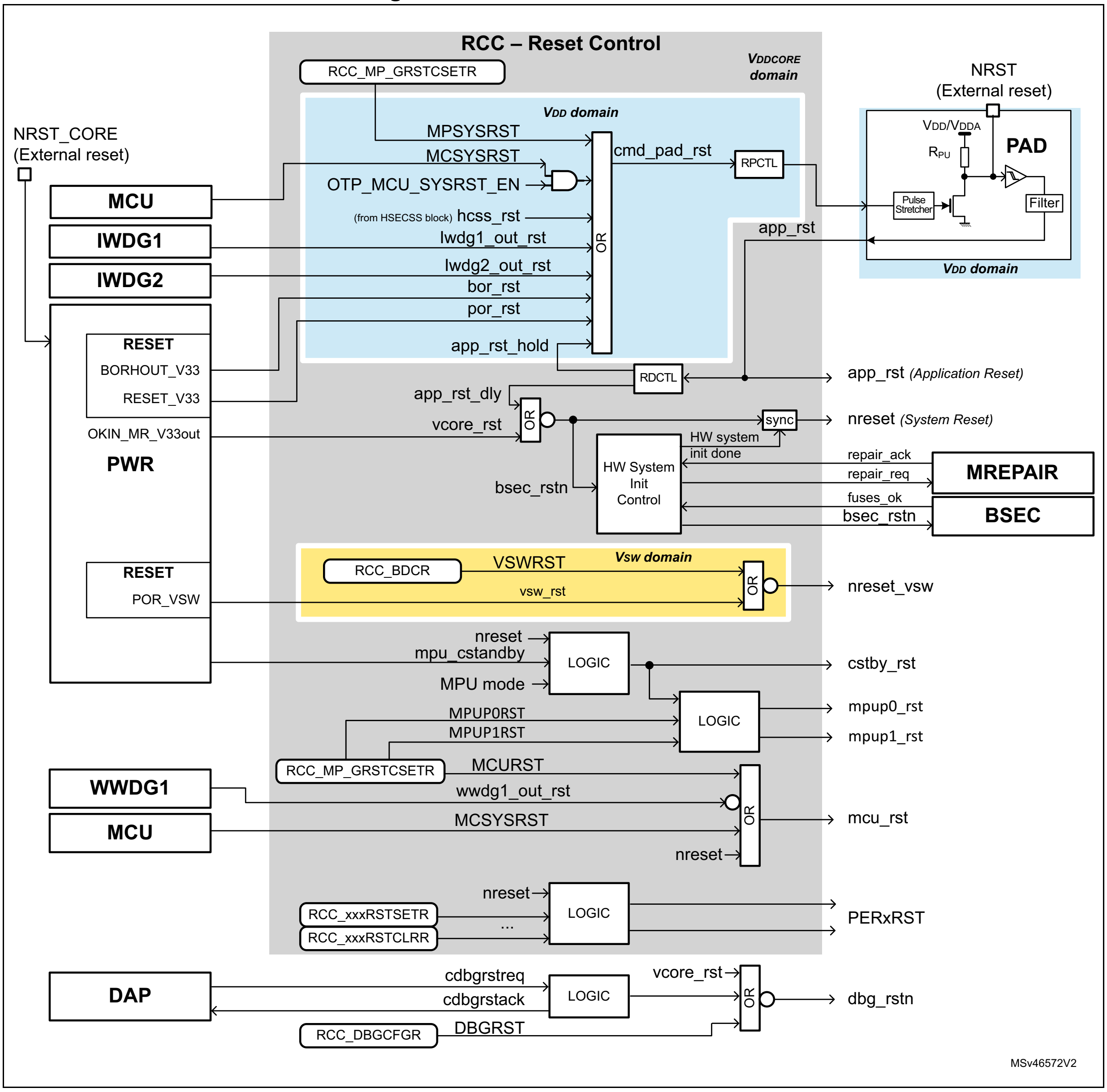
As shown in *Figure 47*, the NRST pad is activated by:

- **por_rst**, **bor_rst**,
- **iwdg1_out_rst**, **iwdg2_out_rst,**
- MPSYSRST,
- MCSYSRST when allowed by OTP_MCU_SYSRST_EN option byte. Refer to *Section 4: OTP mapping (OTP)*,
- Assertion of NRST by an external source.

A programmable reset pulse control (RPCTL) allows the application to define a minimum pulse width for the assertion of the NRST pad. Note that when the RPCTL is disabled, the reset PAD embeds a pulse stretcher which guarantees a minimum reset pulse duration of 20 µs for each internal reset source. Note that **app_rst** and **nreset** are also impacted by this function. For details on RPCTL refer to *Section 10.3.10: Reset pulse control (RPCTL)*.

Note: It is not recommended to let the NRST and NRST_CORE pins unconnected. When it is not used, connect these pins to ground via a capacitor of 10 to 100 nF.

Figure 47. Reset circuit overview



10.3.3 The local resets

The local resets are the resets which do not directly affect both processors. Here is the list of the most important local resets:

- When the MPU exits from CStandby, it is reset by a reset named **cstby_rst**. CStandby mode means that the MPU requested to go to Standby but the product did not go to Standby due to the mode of the MCU. Refer to *Section 9: Power control (PWR)* for details.

- The MPU can reset the MCU via the RCC_MP_GRSTCSETR register, by setting the bit MCURST to ‘1’.
- The MPU can reset the MPU processor 0 via the RCC_MP_GRSTCSETR register, by setting the bit MPUP0RST to ‘1’. Note that the Snoop Control Unit of the MPU (SCU) is not reset.
- The MPU can reset the MPU processor 1 via the RCC_MP_GRSTCSETR register, by setting the bit MPUP1RST to ‘1’. Note that the Snoop Control Unit of the MPU (SCU) is not reset.
- A software reset from the Cortex-M4 via the AIRCR register can be restricted only to the MCU and WWDG1 according to the value of the option byte OTP_MCU_SYSRST_EN.
- A timeout of WWDG1 will reset the MCU and WWDG1.

10.3.4 Resets generated by the CPUs

Both CPUs can generate system and local resets. The RCC offers the following possibilities:

- The MPU can generate a system reset by means of the MPSYSRST bit, located into *RCC global reset control set register (RCC_MP_GRSTCSETR)*. The MCU cannot access the MPSYSRST bit, and it cannot prevent the MPU to generate this system reset.
- The MPU can reset the MCU by means of the MCURST bit, located into *RCC global reset control set register (RCC_MP_GRSTCSETR)*. The MCU cannot prevent the MPU to generate this local reset.
- The MPU can reset its processor 0 or processor 1 independently thanks to MPUP0RST and MPUP1RST bits, located into *RCC global reset control set register (RCC_MP_GRSTCSETR)*. This feature allows the application to restart one of the MPU core without disturbing code execution of the MCU. Note that the Snoop Control Unit of the MPU (SCU) is not reset.
- The MCU can generate a system reset via the AIRCR register of the Cortex-M4, if allowed by the option byte OTP_MCU_SYSRST_EN. If the option byte does not allow, the reset scope is limited to the MCU and WWDG1. In addition an interrupt can be generated to the MPU via the EXTI block.

Warning: It is recommended to set the MCU in CSleep before performing a MCU reset via MCURST bit.

Note: Setting the bit 2 of AIRCR register located into the MCU, activates the SYSRESETREQ signal. This reset is named MCSYSRST in this document.

Note: The SYSRESETREQ signal is connected to both EXTI and GIC allowing the MPU to receive a wakeup event and an interrupt when the MCU generates a MCSYSRST request.

The MPU will be reset either by a system reset (**nreset**) or by a CStandby local reset (**cstby_rst**).

10.3.5 Watchdog reset

Three watchdogs are provided to the application:

- The IWDG1 and IWDG2 dedicated to the MPU
- The WWDG1 dedicated to the MCU

The WWDG1 can generate a MCU local reset if it not acknowledged in the expected time-window. In addition an interrupt can be generated to the MPU via the EXTI block. Refer to *Section 46.2: IWDG main features* for details.

The IWDG1 and IWDG2 generate a system reset when a timeout occurs.

10.3.6 Backup domain reset

A backup domain reset (**nreset_vsw**) is generated when one of the following events occurs:

- Software reset, triggered by setting the VSWRST bit in the *RCC backup domain control register (RCC_BDCR)*. All RTC registers and the RCC_BDCR register are set to their reset values, except the field VSWRST itself. The BKPSRAM and RETRAM are not affected.
- When the V_{SW} voltage is outside the operating range. All RTC registers and the RCC_BDCR register are set to their reset values. In this case the content of the BKPSRAM and RETRAM are no longer valid.

Refer to *Section 1.3.3: Backup domain* section of PWR block for additional information.

10.3.7 Coresight debug reset

The coresight debug components including the debug part of the MPU, can be reset in three different ways:

- Over the DAP by setting to ‘1’ the bit CDBGRSTREQ of the control/status register of the debug port. Setting the bit CDBGRSTREQ to ‘1’ asserts the debug reset request signal CDBGRSTREQ connected to the RCC. Then the RCC activates the **dbg_rstn**, a handshake signal **cdbgrstack** is provided to the DAP request. The **dbg_rstn** is active as long as the **cdbgrstreq** is to ‘1’.
- By the application setting the bit DBGRST of *RCC debug configuration register (RCC_DBGCFGR)* to ‘1’. When DBGRST is set to ‘1’, the **dbg_rstn** reset is activated, the **dbg_rstn** reset is released when DBGRST = ‘0’.
- When V_{DDCORE} power-on reset occurs (**vcore_rst**). This reset is activated after a power-on reset, or when the product exits from Standby.

10.3.8 Option bytes loading

The option bytes configuration and the memory repair sequences are triggered every time the **app_rst** or the **vcore_rst** are activated (i.e. after an application reset, or when the system exits from Standby).

10.3.9 Peripheral reset

The application can reset individually any peripherals whenever it is requested. This can be done by both processors via registers named RCC_xxxxRSTSETR and RCC_xxxxRSTCLRR, “xxxx” is the bus name where the peripheral is connected.

Note that when the secure mode is enabled, only secure accesses can reset the peripherals connected to APB5.

When a peripheral needs to be reset, it is recommended to first enable its peripheral clock by setting the corresponding peripheral enable bit to ‘1’. Refer to *Section 10.4.10: Peripheral allocation* for details.

For other peripherals, it is up to the application to assert and release the peripheral reset (respectively via RCC_xxxxRSTSETR, and RCC_xxxxRSTCLRR registers).

10.3.10 Reset pulse control (RPCTL)

The RPCTL allows the application to control the minimum activation time of the NRST PAD. This feature is particularly helpful because some external devices may request a specific duration for the activation of the NRST. In addition the reset pulse generated by the IWDGx or WWDG, may be too short for external devices.

The RPCTL is located into the V_{DD} domain, and is only reset after a power-on reset.

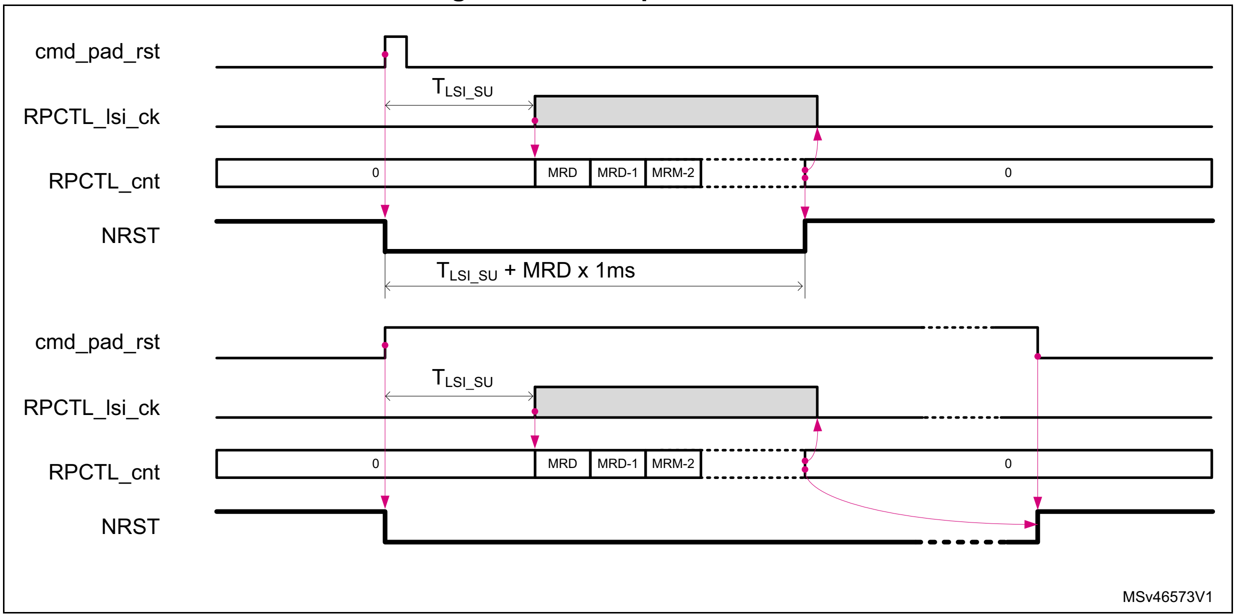
The RPCTL is controlled by the MRD[4:0] bits located in the *RCC reset duration and LSI control register (RCC_RDLSICR)*.

- If MRD = ‘0’, then the RPCTL is bypassed and the signal **cmd_pad_rst** directly drives the reset PAD. The minimum activation time in this case is given by the pulse stretcher embedded into the reset PAD (typically 20 µs).
- If MRD is different from ‘0’, when the reset is activated (rising edge of **cmd_pad_rst**), NRST is immediately driven Low at least for a duration equal to the value given by MRD[4:0]. The duration of the reset is not affected by the RPCTL in the case where the

cmd_pad_rst signal is active for a duration longer than the value programmed on MRD[4:0]. The MRD[4:0] bits allow to adjust a minimum activation time between 1 to 31 ms.

Note that the RPCTL is using the LSI clock in order to control the reset width. When the **cmd_pad_rst** goes HIGH, then the RPCTL requests the LSI clock in order to control the reset duration. If the LSI was not enabled by another function, it may take some microseconds before having the LSI ready (T_{LSI_SU}). *Figure 48* shows simplified timing diagrams.

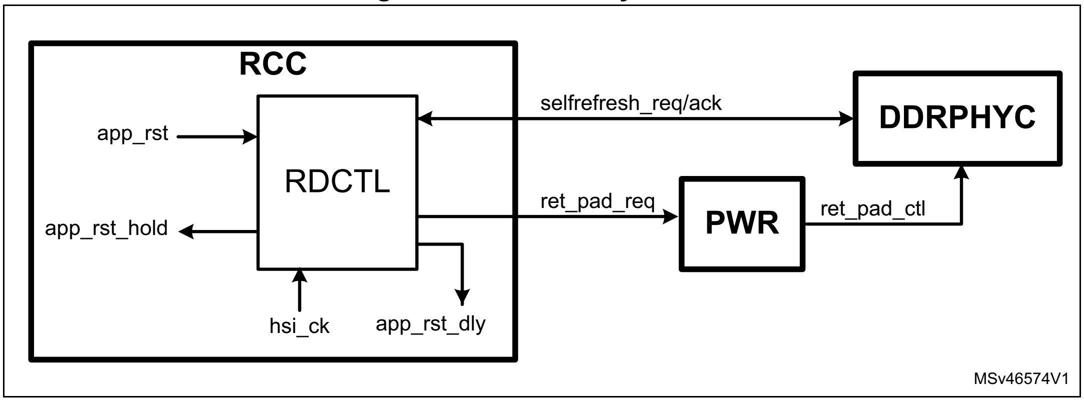
Figure 48. Reset pulse control



10.3.11 Reset delay control (RDCTL)

The RDCTL block must be used if the external DDR device needs to be set in self-refresh before generating a system reset (**nreset**) to the circuit.

Figure 49. Reset delay control



When enabled, the RDCTL performs the following operations:

If a **app_rst** occurs:

- The RDCTL keeps the **app_rst_dly** inactive and asserts the **app_rst_hold** signal in order to maintain the **cmd_pad_rst** active,
- The RDCTL activates the HSI oscillator, and waits if needed, the availability of **hsi_ck** clock,
- If the system is in Stop, LP-Stop or LPLV-Stop mode, then the **app_rst_dly** signal is activated and the **app_rst_hold** signal is released.
- Otherwise, if the system is in Run, then:
  - The timeout counter starts to count-down,
  - The RDCTL requests to the DDRPHYC to set the external DDR device into self-refresh.
  - Once the DDRPHYC confirms that the external DDR device is in self-refresh, then the RDCTL performs the following actions:
    - Freeze of the timeout counter,
    - Requests to the PWR to set the DDR pads into retention mode (**ret_pad_req**),
    - Activates the **app_rst_dly** signal,
    - Releases the **app_rst_hold** signal.
  - If the timeout counter elapses before the RDCTL receives the confirmation that the external DDR device is set to self-refresh, the **app_rst_dly** is activated, and the signal **app_rst_hold** is released.

The RDCTL can be enabled or disabled by the application via the RDCTLEN bit of *RCC application reset control register (RCC_MP_APRSTCR)*. When the RDCTL is disabled (RDCTLEN = ‘0’), the signal **app_rst_dly** is not delayed and follows **app_rst** value, in addition, the DDR pads retention request is not generated.

Caution: The bit RDCTLEN must not be set to ‘1’, at the moment where the **app_rst** is asserted. It is preferable to enable this function during the initialization phase.

The application can also program the timeout value via the RSTTO[6:0] field of *RCC application reset control register (RCC_MP_APRSTCR)*.

This timeout value can be programmed using the following formula:

$$\mathrm{RSTTO} \geq \mathrm{round}\left(\frac{8192 \times T_{ck\_ker\_dphy}}{10^{-6}}\right)$$

Where $T_{ck\_ker\_dphy}$ represents the period of the kernel clock provided to the DDRPHYC.

Caution: It is not allowed to change this field value when RDCTLEN = ‘1’.
It is not recommended to set RSTTO[6:0] to ‘0’.

After a **nreset**, the application can check if the external DDR device has been properly set in self-refresh by checking the RSTTO[6:0]. The RSTTO[6:0] gives the current value of the timeout counter. If this value is equal to ‘0’, it means that a timeout condition occurs at the last time the RDCTL has requested to the external DDR device to self-refresh.

*Note: The **hsi_ck** frequency depends on the HSIDIV value.*

10.3.12 Reset coverage summary

Table 50 gives a detailed view of reset sources and reset domains (highlighted in gray). Each line represents a reset source.

Note: It is assumed that when V_{DD} is not valid, V_{DDCORE} is not valid as well.

Table 50. Reset coverage summary (1)

Reseted functions		Reset lines											
		por_rst (2)	vcore_rst (3)	app_rst (4)	nreset	cstby_rst	mpup1_rst	mpup0_rst	mcu_rst	dbg_rstn	bsec_rstn	PERxRST	nreset_vsw
V_{DD} domain		x	-	-	-	-	-	-	-	-	-	-	-
DDR ITF.		x	x	x	x	x	-	-	-	-	-	-	-
MPU processor 0		x	x	x	x	x	-	x	-	-	-	-	-
MPU processor 1		x	x	x	x	x	x	-	-	-	-	-	-
MCU and WWDG1		x	x	x	x	-	-	-	x	-	-	-	-
AXI/AHB Interconnections.		x	x	x	x	-	-	-	-	-	-	-	-
Debug components, including DBGMCU Reset all the debug parts including the ones located into the MPU core except the SWJ-DP function. The SWJ-DP is reset by the nTRST or **vcore_rst** resets.		x	x	x	-	-	-	-	-	x	-	-	-
IWDG[2:1]		x	-	x	-	-	-	-	-	-	-	-	-
Hardware system init. It includes the reload of the option bytes, and the memory repair.		x	x	x	x	-	-	-	-	-	x	-	-
RCC	RCC_MP_BOOTCR	x	-	x	-	-	-	-	-	-	-	-	-
	RCC_BR_RSTSCLRR, RCC_MC_RSTSCLRR	x	-	-	-	-	-	-	-	-	-	-	-
	RCC_MP_IWDGFZSETR	x	x	x	x	x	-	-	-	-	-	-	-
	RCC_RDLSICR	x	-	-	-	-	-	-	-	-	-	-	-
	RCC_BDCR	-	-	-	-	-	-	-	-	-	-	-	x
	RCC_MP_GCR	x	x	x	x	-	-	-	-	-	-	-	-
	Other RCC registers	x	x	x	x	-	-	-	-	-	-	-	-
PWR	PWR_CR2	-	-	-	-	-	-	-	-	-	-	-	x
	PWR_CR3	x	-	-	-	-	-	-	-	-	-	-	-
	PWR_MPUCR, PWR_MCUCR Individual bits of these registers do not have the same reset condition, refer to the PWR section for details.	x	x	x	x	-	-	-	-	-	-	-	-
	PWR_WKUPCR, PWR_WKUPFR, PWR_MPUWKUPENR, PWR_MCUWKUPENR	x	-	x	-	-	-	-	-	-	-	-	-
	Other registers	x	x	x	x	-	-	-	-	-	-	-	-
RTC		-	-	-	-	-	-	-	-	-	-	-	x

Table 50. Reset coverage summary (continued)[(1)]

Reseted functions	Reset lines											
	por_rst [(2)]	vcore_rst [(3)]	app_rst [(4)]	nreset	cstby_rst	mpup1_rst	mpup0_rst	mcu_rst	dbg_rstn	bsec_rstn	PERxRST	nreset_vsw
BKPSRAM and RETRAM After a reset of the V_{SW} domain, the backup regulator of the BKPSRAM and RETRAM are disabled, this function is controlled via the PWR block, BREN and RREN bits. If the nreset_vsw reset is due to a too low V_{SW} voltage, the content of BKPSRAM and RETRAM is lost.	-	-	-	-	-	-	-	-	-	-	-	x
Other Peripherals	x	x	x	x	-	-	-	-	-	-	x	-

1. The ‘x’ means that the function is reset by the corresponding reset line, the ‘-’ means that the function is not reset by the corresponding reset line.
2. When **por_rst** is asserted, the **vcore_rst**, the **nreset**, the **app_rst**, the **bsec_rst,** the **mcu_rst** are asserted as well.
3. When **vcore_rst** is asserted, the **nreset**, the **bsec_rst,** the **mcu_rst** the **dbg_rst** are asserted as well.
4. When **app_rst** is asserted, the **nreset**, the **bsec_rst**, the **mcu_rst** are asserted as well.

Reset of the RCC registers

Table 51 lists registers having bits into a voltage domain different from V_{DDCORE}. As a consequence some of those registers are not always reset by **nreset**. The RCC registers not shown in *Table 51* are reset by the system reset (**nreset**).

Table 51. RCC registers with specific reset and specific voltages

RCC register name	Voltage domain	Reset condition	Comment
RCC_MP_BOOTCR	V_{DD}	app_rst	Shall be reset after a System reset, but not after Standby
RCC_OCENSETR, RCC_OCENCLRR, RCC_OCRDYR, RCC_HSICFGR, RCC_CSICFGR	V_{DD}	nreset	-
RCC_BR_RSTSCLRR	V_{DD}	por_rst	Only reset when a V_{DD} is removed
RCC_MC_RSTSCLRR	V_{DD}	por_rst	Only reset when a V_{DD} is removed
RCC_RDLSICR	V_{DD}	por_rst	Only reset when a V_{DD} is removed
RCC_BDCR	V_{SW}	nreset_vsw	-

10.3.13 Reset source identification

The RCC offers four registers in order to identify the root cause of resets:

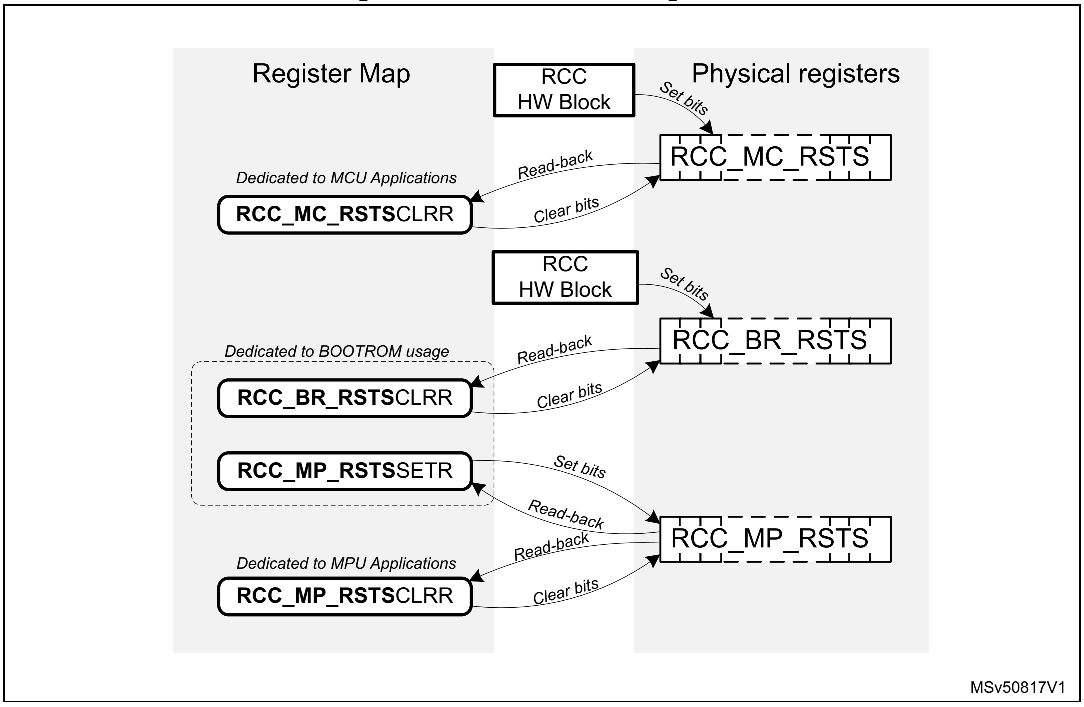
- *RCC BOOTROM reset status clear register (RCC_BR_RSTSCLRR)*,
- *RCC MCU reset status clear register (RCC_MC_RSTSCLRR),*
- *RCC MPU reset status clear register (RCC_MP_RSTSCLRR)* and
- *RCC MPU reset status set register (RCC_MP_RSTSSETR)*

The registers RCC_MP_RSTSSETR and RCC_BR_RSTSCLRR are dedicated to the BOOTROM usage, and must not be used by the application.

The register RCC_MP_RSTSCLRR is dedicated to the MPU applications, and the register RCC_MC_RSTSCLRR is dedicated to the MCU applications.

The figure hereafter is giving details on the way the reset source registers are working.

Figure 50. Reset source registers



The BOOTROM will use the RCC_BR_RSTSCLRR register in order to identify the root cause of the reset, and to define which kind of boot sequence must be performed. In addition the BOOTROM code will update the RCC_MP_RSTSCLRR register after every system reset, power-on reset or exit from Standby and CStandby. More precisely, the BOOTROM copies the content of RCC_BR_RSTSCLRR and the value of SBF and SBF_MPU flags into the RCC_MP_RSTSCLRR register, using the RCC_MP_RSTSSETR register.

Then the MPU can check and clear the reset sources by reading the RCC_MP_RSTSCLRR register. On its side, the MCU can check and clear the reset sources by reading the RCC_MC_RSTSCLRR register. Each CPU can reset the flags of their own registers.

Table 52 shows how the status bits of RCC_MC_RSTSCLRR, RCC_BR_RSTSCLRR and RCC_MP_RSTSCLRR registers behave, according to the situation generating a reset.

Table 52. Reset source identification (RCC_MP/MC/BR_RSTSCLRR, and PWR)[1]

#	Situations Generating a Reset	HCSSRSTF	WWDG1RSTF	IWDG2RSTF	IWDG1RSTF	CSTDBYRSTF [2]	STDBYRSTF [2]	MCSYSRSTF	MPUP1RSTF [3]	MPUP0RSTF [3]	MPSYSRSTF	PORRSTF	BORRSTF	VCORERSTF	PADRSTF	MCURSTF
1	Power-on reset (**por_rst**)	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
2	Brownout reset (**bor_rst**)	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
3	Pad reset from NRST	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
4	System reset generated by MCU (MCSYSRST), when system reset is allowed by the option byte OTP_MCU_SYSRST_EN.	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
	Local reset generated by MCU (MCSYSRST), when system reset is not allowed by the option byte OTP_MCU_SYSRST_EN.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
5	System reset generated by MPU (MPSYSRST)	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
6	MCU reset by MPU (MCURST)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
7	System exits from Standby	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	MPU exits from CStandby	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
9	MPU processor 0 reset generated by the MPU	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
10	MPU processor 1 reset generated by the MPU	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
11	WWDG1 reset (**wwdg1_out_rst**)	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
12	IWDG1 reset (**iwdg1_out_rst**)	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
13	IWDG2 reset (**iwdg2_out_rst**)	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
14	Reset due to a failure of V_{DDCORE} or assertion of NRST_CORE	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	Reset due to a clock failure on HSE	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1

1. The bit status at ‘1’ is highlighted in gray.
2. Those bits are located into the CC_MP_RSTSCLRR and RCC_MP_RSTSSETR registers. The BOOTROM copies the SBF bit into STDBYRSTF and the SBF_MPU bit into CSTDBYRSTF. The SBF and SBF_MPU are described into *Section 1.9.5: PWR MPU control register (PWR_MPUCR)*.
3. Bits MPUP[1:0]RSTF are not available into the RCC_MC_RSTSCLRR register.

Note: The MCSYSRSTF bit located into the RCC_BR_RSTSCLRR *and* RCC_MP_RSTSR *registers is only meaningful when the option byte OTP_MCU_SYSRST_EN allows the MCU to perform a system reset. If it is not the case the bit MCSYSRSTF is forced to ‘0’.*

10.3.14 Power-on and wakeup sequences

For detailed diagrams refer to *Section 1.3.1: Supply system startup* of the PWR block.

The time interval between the event which exits the product from a low-power mode and the moment where the CPUs are able to execute code, depends on the state of the system and its configuration. *Figure 51* shows the most usual examples.

Power-on wakeup sequence

The “power-on wakeup” sequence of *Figure 51* gives the most significant phases of the power-on sequence. It is the longest sequence as the circuit was not powered. Note that this sequence remains unchanged if VBAT was present or not.

Boot sequence from PAD reset (NRST)

The “PAD reset” gives the most important steps of the boot sequence. Note that in this case, V_{DD}/V_{DDA} have not been removed.

- The setting time is faster as the reference voltage is already stable.
- The HSI restart delay may be needed if the HSI is not enabled when the NRST occurs, otherwise this restart delay phase is skipped.
- The option byte reloading is performed.

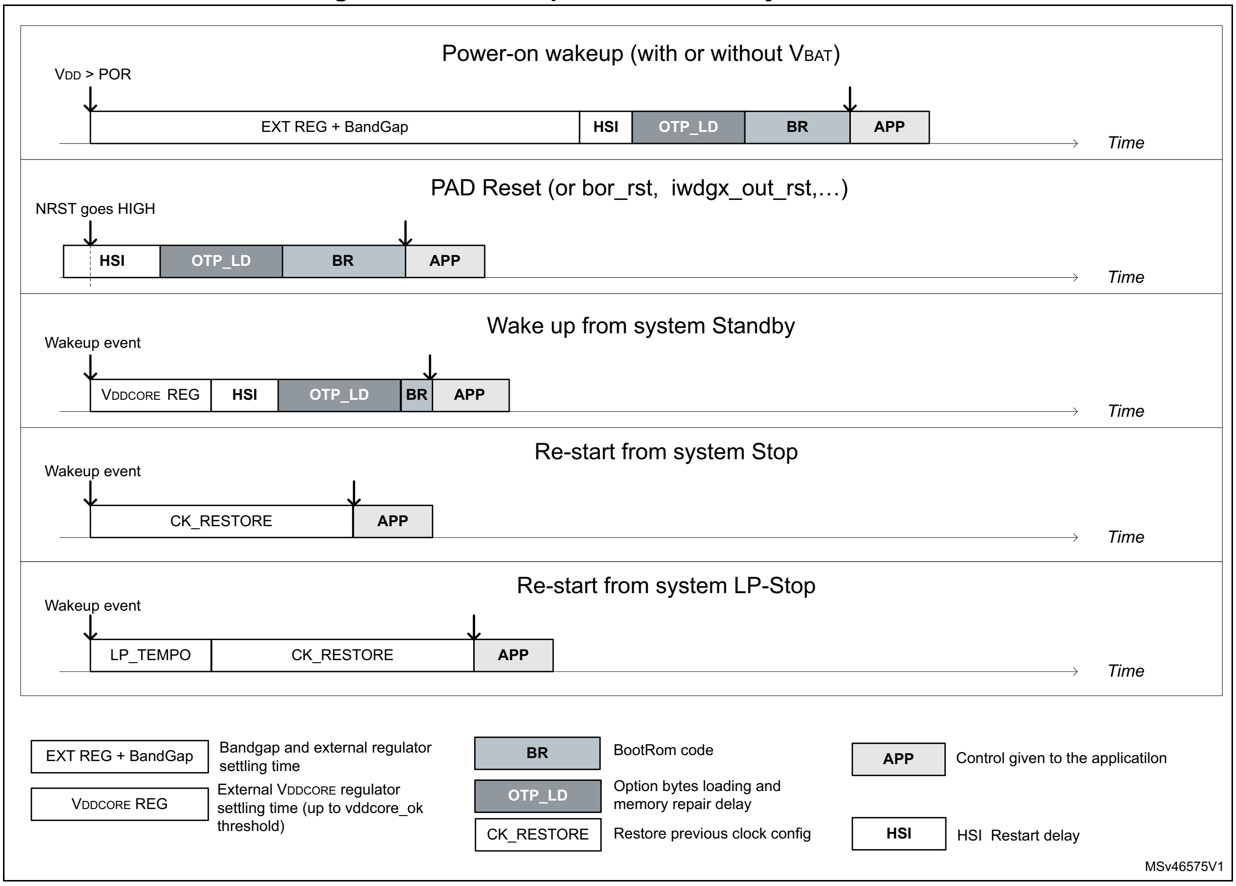
Boot sequence from system Standby

For a wake up from system Standby, as the V_{DD} is not removed, the settling time is also fast as the reference voltage is already stable. As the V_{DDCORE} was not there, the re-start delay for the HSI, and the option byte reloading cannot be skipped.

Re-start sequence from system Stop

For a re-start from system Stop, V_{DDCORE} has not been removed, so the settling time is mainly due to the re-start delay of the clock. It can be fast if the system has been programmed to restart on HSI/CSI, but it will be longer if the system restarts on HSE, and/or PLLs.

Figure 51. Boot sequences versus system states



MCU HOLD_BOOT after processor reset

The RCC allows to put the MCU on HOLD_BOOT every time the MCU is reset.

This function is controlled by the MPU via the BOOT_MCU bit located into the *RCC global control register (RCC_MP_GCR)*. The BOOT_MCU bit can only be programmed by the MPU. In addition, if the secure mode is activated, only secure MPU accesses will be allowed.

When BOOT_MCU = ‘0’, after a MCU reset, the MCU is maintained on HOLD_BOOT until the MPU sets the bit BOOT_MCU to ‘1’.

The possible reset sources for the MCU are:

- System reset (**nreset**),
- The reset generated by the MPU (MCURST),
- The WWDG1 reset (wwdg1_out_rst),
- The reset generated by the MCU itself (MCSYSRST).

Having the MCU on HOLD_BOOT, does not prevent the system to go to one of the Stop modes, or in Standby. So when the MCU is on HOLD_BOOT, the system will be in Stop mode if the MPU goes to CStop, then if the MPU allows the Standby mode, the system can go to Standby (if PDDS bits are set to ‘1’).

In conclusion, the HOLD_BOOT mode is equivalent to a MCU CStop mode. This means that when the MCU is in HOLD_BOOT, all the peripherals allocated by the MCU, will no longer be clocked, except the ones able to request a kernel clock, or working with **lse_ck** or **lsi_ck**.

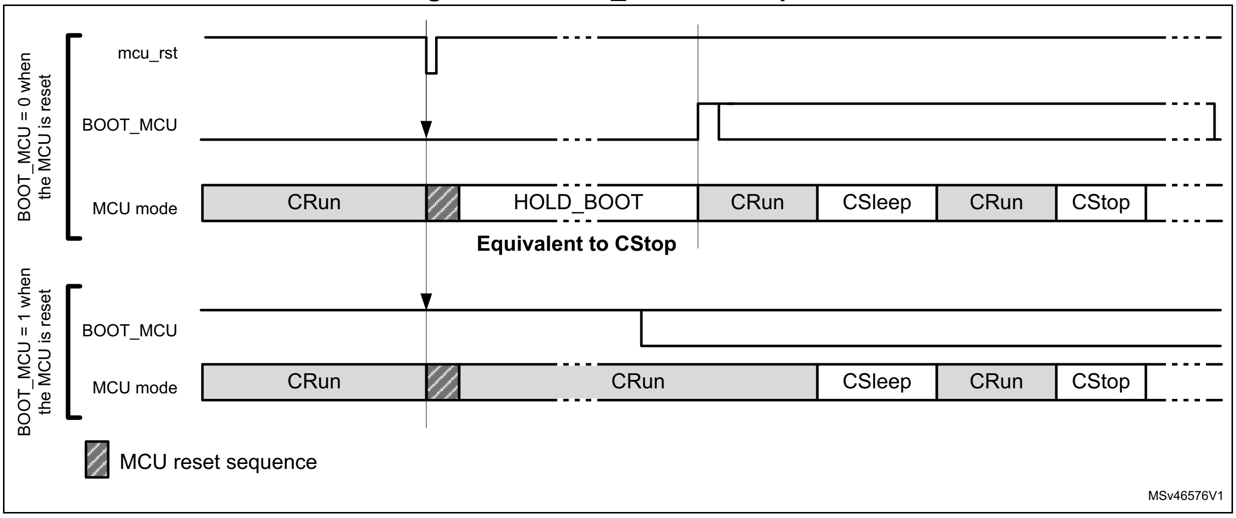
The BOOT_MCU bit is reset after a system reset, a power-on reset, or an exit from Standby (i.e. **nreset** signal).

Figure 52 shows two examples. An example where the BOOT_MCU bit is set to ‘0’, when the MCU is reset. In that case the MCU stays in HOLD_BOOT mode until the MPU sets the BOOT_MCU bit to ‘1’.

In the second example, the bit BOOT_MCU bit is set to ‘1’, when the MCU is reset. In that case the MCU switches immediately to CRun mode as soon as the MCU hardware reset sequence is completed.

Note that once the MCU is in CRun, the BOOT_MCU can be set again to ‘0’ without disturbing the MCU activity. In that way the MCU will be again set in HOLD_BOOT the next time it is reset.

Figure 52. HOLD_BOOT examples



Processor hold from Standby

The BOOTROM code provides the possibility to select which processor is allowed to restart its application code after a Standby mode. This feature is controlled via two bits: MPU_BEN and MCU_BEN.

The bits MPU_BEN and MCU_BEN can only be programmed by the MPU. In addition, if the secure mode is activated, only secure MPU accesses can change those bits. They are reset by hardware after a system reset (from IWDG, POR, BOR, PAD), but not when the product exits from Standby. Those bits are located into V_{DD} domain in order to be reset after a power-on reset. Refer to the *RCC boot control register (RCC_MP_BOOTCR)*.

After a system reset, a power-on reset, or an exit from Standby, the MCU is always maintained on HOLD_BOOT by the hardware (because BOOT_MCU bit is reset), and only the MPU is allowed to boot, thus to execute the BOOTROM code.

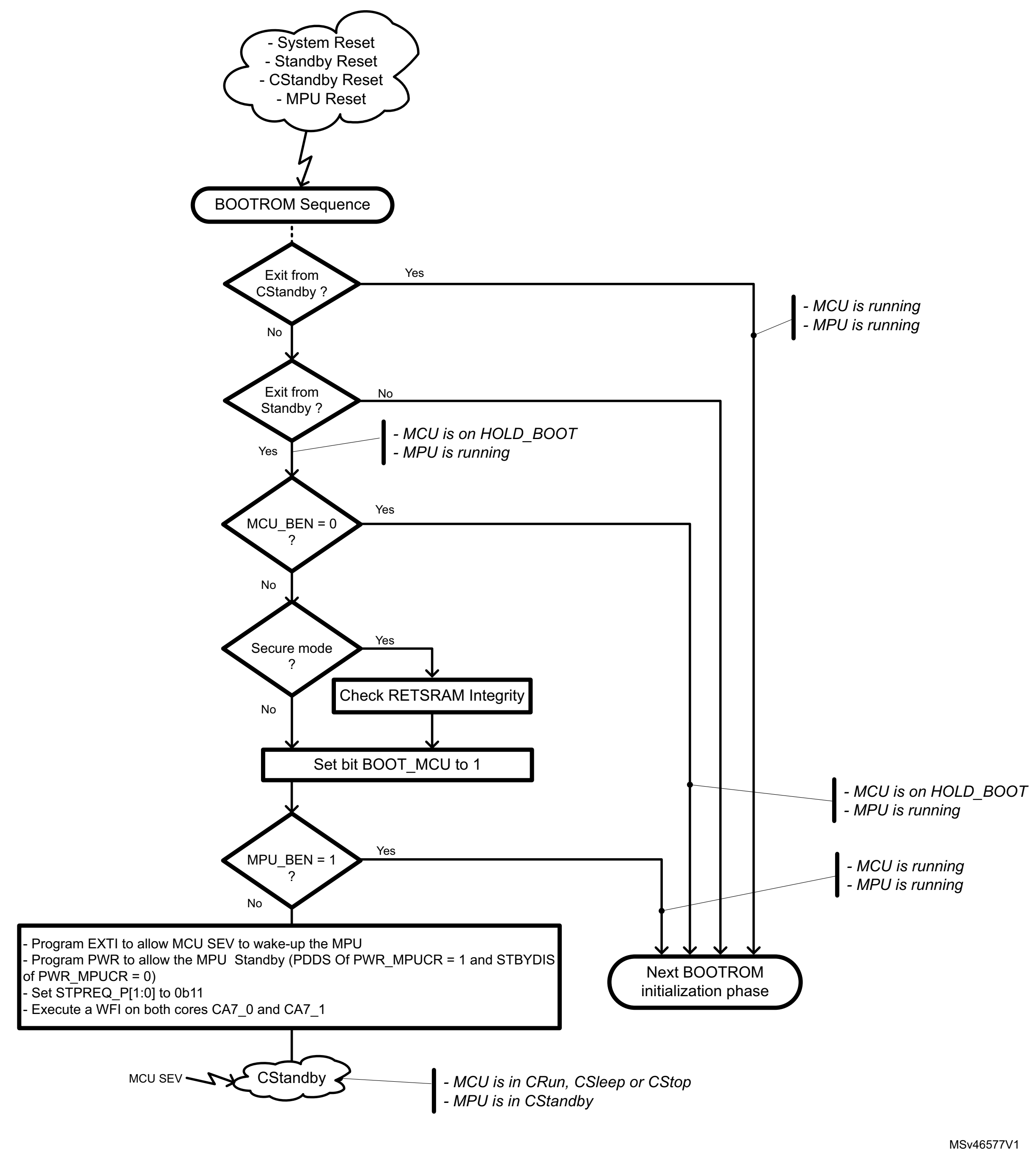
If the application wants that only the MCU restarts after a product Standby, the MPU shall set the bit MCU_BEN to ‘1’, and MPU_BEN to ‘0’ before the product goes to Standby. Then when the product exits from Standby, the BOOTROM will check the integrity of the RETRAM code (only in secure mode), and give the control to the MCU, by setting the bit BOOT_MCU to ‘1’.

Then the BOOTROM sets the MPU to CStandby. The MPU can exit from the CStandby mode by a MCU SEV event, or a system reset.

If MPU_BEN was set to ‘1’, then the BOOTROM continue its initialization phase and jumps to the application instead of going to CStandby.

Note: The SEV is a Cortex-M4 instruction which can be used to generate a pulse event on the TXEV line. This line is connected to the EXTI in order to generate an event for the MPU even if the MPU is in CStop or CStandby.

Figure 53. Hold function handled by the BOOTROM



10.4 RCC functional description - clock part

The RCC provides a high degree of flexibility to the application in the choice of the clock generators:

- The clock from HSI: High Speed Internal oscillator (~ 8, 16, 32, 64 MHz)
- The clock from HSE: High Speed External oscillator (8 to 48 MHz)
- The clock from LSE: Low Speed External oscillator (32.768 kHz)
- The clock from LSI: Low Speed Internal oscillator (~ 32 kHz)
- The clock from CSI: Low Power Internal oscillator (~4 MHz)

It offers a good flexibility for the application to select the appropriate clock for CPUs and peripherals. More especially for peripherals that need a specific clock like Ethernet, USB OTG FS and HS, SPI(I2S), SAI and SDMMC.

Each clock source can be switched on or off independently in order to optimize the power consumption.

There are mainly 4 clock paths:

- The MPU clocking
- The AXI Sub-System (AXI-SS) clocking, including the DDR interface
- The MCU, and its bus matrix
- The peripheral kernel clocks

The RCC provides up to 4 PLLs, each of them can be configured with integer or fractional ratios.

- The PLL1 is dedicated to the MPU clocking
- The PLL2 provides:
  - The clocks for the AXI-SS (including APB4, APB5, AHB5 and AHB6 bridges)
  - The clocks for the DDR interface
- The PLL3 provides:
  - The clocks for the MCU, and its bus matrix (including the APB1, APB2, APB3, AHB1, AHB2, AHB3 and AHB4)
  - The kernel clocks for peripherals
- The PLL4 is dedicated to the generation of the kernel clocks for various peripherals

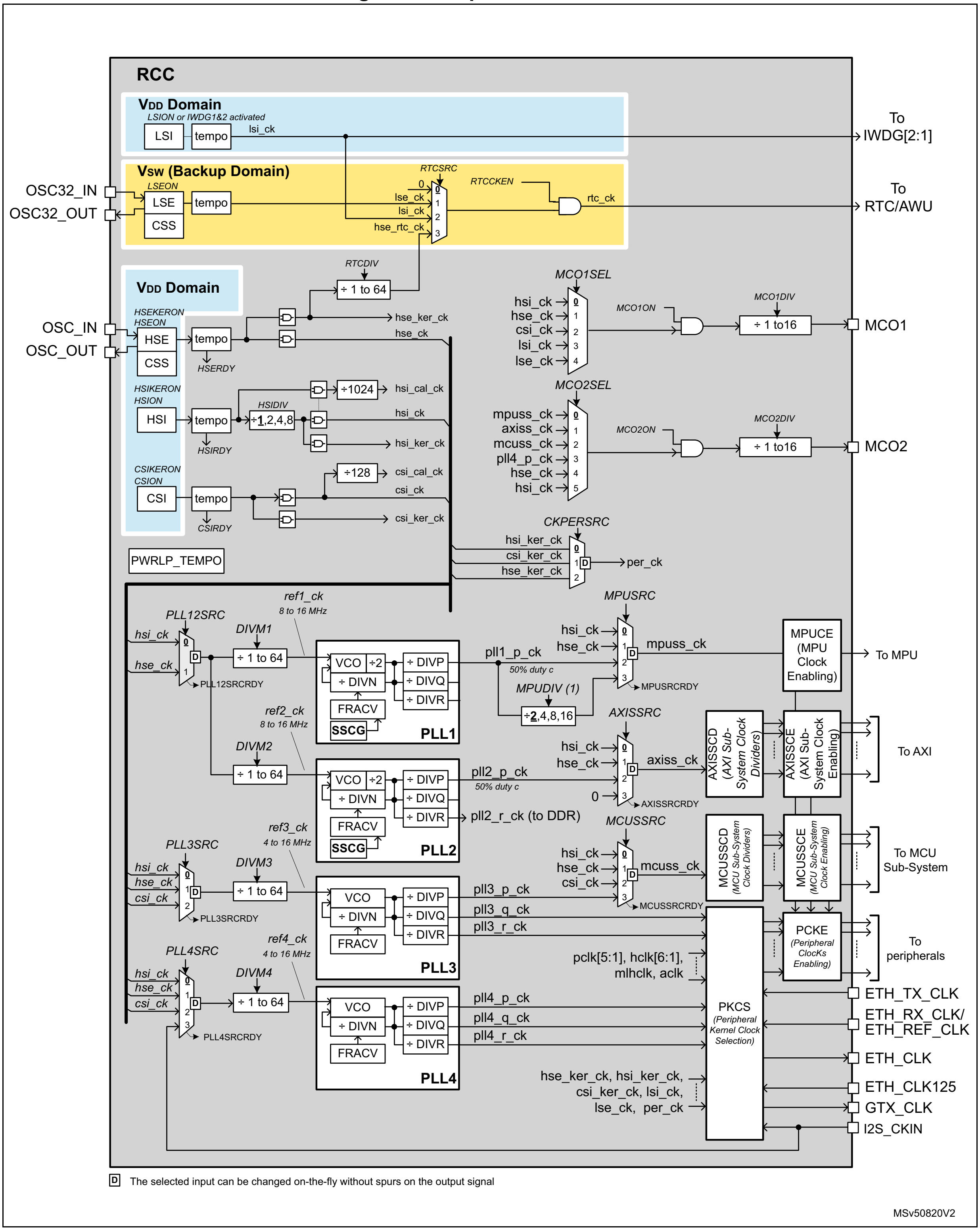
The blocks named AXISSCD, and MCUSSCD contain several prescalers used to configure the processors and bus matrix clock frequencies.

The blocks named AXISSCE, MCUSSCE and PCKE are responsible for the gating of the peripheral kernel clock, the bus interface, the cores and bus matrix clocks.

The block named PKCS (Peripheral Kernel Clock Selection) provides several dynamic switches allowing a large choice for the kernel clock distribution to peripherals.

As shown in *Figure 54*, the RCC offers 2 clock outputs (MCO1 and MCO2), with a great flexibility on the clock selection and frequency adjustment.

Figure 54. Top-level clock tree



2. **X** represents the selected MUX input after a system reset.

10.4.1 Clock naming convention

The RCC is providing clocks to the complete circuit. In order to avoid misunderstanding, the following terms are used in this document:

- **The peripheral clocks:**
  The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clocks can be provided to a peripheral:
  - The bus interface clocks
  - The kernel clocks

  A peripheral will receive from the RCC one or several bus interface clocks. This clock is generally the AHB, APB or AXI clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (example GPIOx, DMAx...).

  Some peripherals request as well a dedicated clock in order to handle the interface function. This clock is named kernel clock. For example peripherals such as SAI, need to generate specific and accurate master clock frequencies, which requests dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific need of the interface, is that the bus clock can be changed without requesting any reprogramming of the peripheral.
- **The CPU clocks**
  The CPU clock is the clock provided to the CPUs (**mcu_ck**, **mpuss_ck**).
- **The bus matrix clocks**
  The bus matrix clocks are the clocks provided to the different bridges (APB, AHB or AXI).

10.4.2 Oscillators description

The table hereafter shows the oscillator states versus system modes when the oscillators are enabled via registers. The term “available” means that the resource can be used if activated via registers.

Table 53. Oscillator states versus system modes

System modes	V_{DD} domain				V_{SW} domain
	HSE	HSI	CSI	LSI	LSE
Exit from system reset	OFF	ON	OFF	Available	Available
In Run	Available	Available	Available	Available	Available
In Stop, LP-Stop	Available (1)	Available (2)	Available (3)	Available	Available
In Standby, LPLV-Stop	OFF	OFF	OFF	Available	Available
In VBAT	OFF	OFF	OFF	OFF	Available

1. HSE can remain activated in (LP-)Stop mode if HSEKERON = ‘1’, or if the peripheral generates a kernel clock request.
2. HSI can remain activated in (LP-)Stop mode if HSIKERON = ‘1’, or if the peripheral generates a kernel clock request.
3. CSI can remain activated in (LP-)Stop mode if CSIKERON = ‘1’, or if the peripheral generates a kernel clock request.

The HSE oscillator

The High Speed External block can generate a clock from two possible sources:

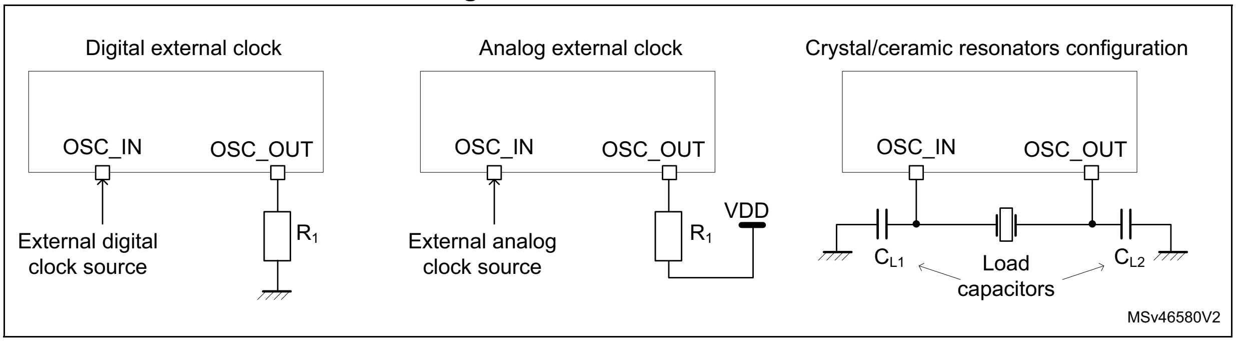
- external analog clock source
- external digital clock source
- external crystal/ceramic resonator

In order to allow the BOOTROM to detect in which configuration the HSE is used, a resistor (R1) must be connected either to the GND or to V_{DD}.

The resistor must be connected to the GND in the case where the HSE is using an external digital clock. The resistor must be connected to the V_{DD} in the case where the HSE is using an external analog clock, and removed if a crystal or ceramic resonator is used.

Refer to the datasheet for the values of C_{L1}, C_{L2} and R1.

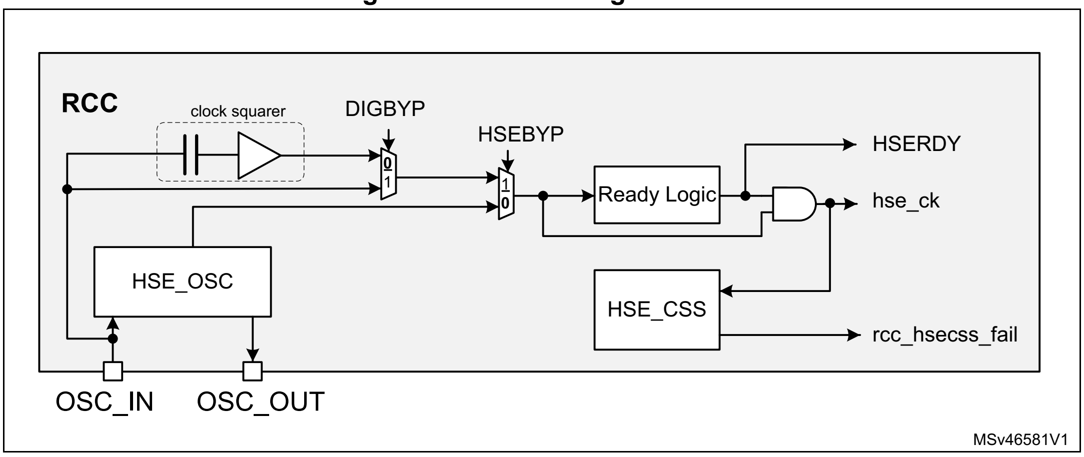
Figure 55. HSE clock source



The HSERDY flag of the *RCC oscillator clock ready register (RCC_OCRDYR)* indicates whether the HSE oscillator is stable or not. When the HSE is enabled (HSEON set to '1'), the HSERDY flag goes to '1' when 512 cycles of HSE clock have been counted. The **hse_ck** clock is not released until the HSERDY bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_MP_CIER)* or *RCC clock source interrupt enable register (RCC_MC_CIER)*.

Figure 56 shows a simplified view of the HSE clock generation.

Figure 56. HSE clock generation



Note as well that it is possible to keep the HSE enabled, when the system is in (LP-)Stop mode, see *Section 10.4.7: Clock generation in Stop and Standby modes* for additional information.

In addition, it is possible to drive the HSE clock to the MCO1 and MCO2 outputs and use it as a clock source for other application components.

External clock configuration (HSE bypass)

The HSE can accept an external clock source on OSC_IN when the bypass mode is selected (HSEBYP = ‘1’).

The external clock signal connected to OSC_IN pin can be digital or low-swing (analog). The low-swing clock signal is supported thanks to an internal clock squarer. The input signal shall have a duty cycle close to 50%.

In order to switch the HSE in external clock mode, the user shall do the following:

- Disable the HSE by writing to ‘1’ the HSEON bit of the *RCC oscillator clock enable clear register (RCC_OCENCLRR)*.
- Check that the HSE is disabled by verifying that the bit HSERDY of *RCC oscillator clock ready register (RCC_OCRDYR)* is set to ‘0’.
- Either set the bit DIGBYP to ‘1’ if the clock provided to OSC_IN is full-swing digital signal, or set the bit DIGBYP to ‘0’ if the clock provided to OSC_IN is low-swing signal. The bit DIGBYP can be controlled via *RCC oscillator clock enable set register (RCC_OCENSETR)* and *RCC oscillator clock enable clear register (RCC_OCENCLRR)*.
- Set the bit HSEBYP to ‘1’ by writing the HSEBYP bit of *RCC oscillator clock enable set register (RCC_OCENSETR)*.
- Enable again the HSE by writing to ‘1’ the bit HSEON of *RCC oscillator clock enable set register (RCC_OCENSETR)*, in order to activate the hse_ck.
- Check that the bit HSERDY of *RCC oscillator clock ready register (RCC_OCRDYR)* is set to ‘1’, then the HSE is ready for use.

See *Figure 55* and *Figure 56* for details. Refer to the datasheet for additional information.

External crystal/ceramic resonator configuration

The oscillator is enabled by setting the HSEBYP bit to ‘0’, and HSEON to ‘1’.

The HSE can be used when the product needs a very accurate high speed clock.

In order to select the external crystal/ceramic resonator, the application must follow the sequence hereafter:

- Disable the HSE by writing to ‘1’ the HSEON bit of the *RCC oscillator clock enable clear register (RCC_OCENCLRR)*.
- Check that the HSE is disabled by verifying that the bit HSERDY is set to ‘0’,
- Set the bit HSEBYP to ‘0’, by writing to ‘1’ the HSEBYP bit of *RCC oscillator clock enable clear register (RCC_OCENCLRR)*.
- Enable again the HSE by writing to ‘1’ the bit HSEON, of *RCC oscillator clock enable set register (RCC_OCENSETR)*.
- Check that the bit HSERDY is set to ‘1’, then the HSE is ready for use.

The associated hardware configuration is shown in *Figure 55*. The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize the output distortion and the start-up stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSI oscillator

The HSI provides the default clock to the product.

The HSI is a High Speed Internal RC oscillator which can also be used directly as a source for processors and interconnects clocks, peripheral clock, or as PLL input.

A pre-divider allows the user to select a HSI output frequency of 8, 16, 32 or 64 MHz. This pre-divider is controlled by HSIDIV located into the *RCC HSI configuration register (RCC_HSICFGR)*. A flag (HSIDIVRDY) is also available into the *RCC oscillator clock ready register (RCC_OCRDYR)* in order to check when the new division ratio is taken into account by the hardware.

Note that the HSIDIV value can be changed on-the-fly, and the HSI block will take into account the new division ratio without generating any spurs on the HSI clock.

However the following points must be considered:

- It is **not allowed** to change HSIDIV, if the HSI is currently used as reference clock for a PLL.
- If the HSI clock is currently used as kernel clock for some peripherals, the application has to ensure that the HSI frequency change will not disturb the peripherals.

The HSI has some advantages:

- No need of external crystal; could be interesting as low-cost clock source,
- Startup time faster than HSE: few microseconds,
- Reduced power consumption

The HSI precision, even with the frequency calibration is less accurate than an external crystal oscillator or ceramic resonator.

Care must be taken when the HSI is used as kernel clock for communication peripherals: the application has to take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency accuracy.

Note as well that it is possible to keep the HSI enabled, when the system is in (LP-)Stop mode, see *Section 10.4.7: Clock generation in Stop and Standby modes* for additional information.

The HSIRDY flag of RCC_OCRDYR register indicates whether the HSI is stable or not. At startup, the HSI output clock is not released until the HSIRDY bit is set by hardware. The HSI activation can also be controlled by the application via HSION bit located into the *RCC oscillator clock enable set register (RCC_OCENSETR)* and *RCC oscillator clock enable clear register (RCC_OCENCLRR)*.

In addition, it is possible to drive the HSI clock to the MCO1 and MCO2 outputs and use it as a clock source for other application components.

Calibration

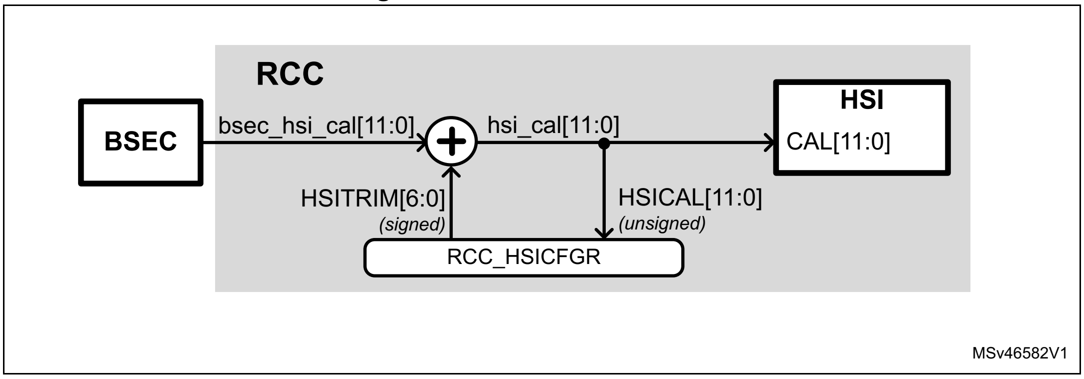
RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for an accuracy of ACC_{HSI}. Refer to the datasheet of the product for more information.

When a **app_rst** or **vcore_rst** occurred, the factory calibration value is reloaded in the HSI, and is visible on HSICAL[11:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user can trim the HSI frequency using the HSITRIM[6:0] bits.

Bits HSICAL[11:0] and HSITRIM[6:0] are located into the *RCC HSI configuration register (RCC_HSICFGR)*.

Figure 57. HSI calibration flow



In addition the RCC provides a signal named **hsi_cal_ck** dedicated to the calibration of the HSI. This signal is connected to TIM15 and TIM12, refer to *Section 10.6.4: The clock calibration using TIMx* for additional information.

Note as well that the signal **hsi_cal_ck** is gated when the system goes to Stop.

The CSI oscillator

The CSI is a Low Power RC oscillator which can also be used directly as a source for processors and interconnects clocks, peripheral clock, or as PLL input for PLL3 and PLL4.

The CSI has some advantages:

- No need of external crystal; could be interesting as low-cost clock source,
- Startup time faster than HSE: few microseconds,
- Very low-power consumption,

Note that the CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

The CSI frequency precision, even with calibration of the frequency, is less accurate than an external crystal oscillator or ceramic resonator.

Even if the settling time of the CSI is faster than the one of the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application has to take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency accuracy.

Note as well that it is possible to keep the CSI enabled, when the system is in (LP-)Stop mode, see *Section 10.4.7: Clock generation in Stop and Standby modes* for additional information.

In addition, it is possible to drive the CSI clock to the MCO1 output and use it as a clock source for other application components.

The CSI can also be controlled by the application via CSION bit located into the *RCC oscillator clock enable set register (RCC_OCENSETR)* and *RCC oscillator clock enable clear register (RCC_OCENCLRR)*.

Calibration

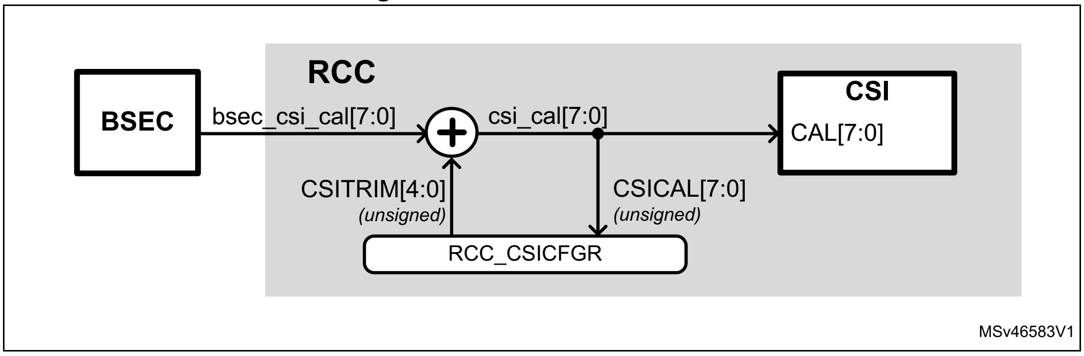
RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by ST for an accuracy of ACC_{CSI}. Refer to the datasheet of the product for more information.

When a **app_rst** or **vcore_rst** occurred, the factory calibration value is reloaded into the CSI, and is visible on CSICAL[7:0] bits.

If the application is subject to voltage or temperature variations this may affect the RC oscillator frequency. The user can trim the CSI frequency using the CSITRIM[4:0] bits.

Bits CSICAL[7:0] and CSITRIM[4:0] are located into the *RCC CSI configuration register (RCC_CSICFGR)*.

Figure 58. CSI calibration flow



In addition the RCC provides a signal named **csi_cal_ck** dedicated to the calibration of the CSI. This signal is connected to TIM15 and TIM12, refer to *Section 10.6.4: The clock calibration using TIMx* for additional information.

Note as well that the signal **csi_cal_ck** is gated when the system goes to Stop.

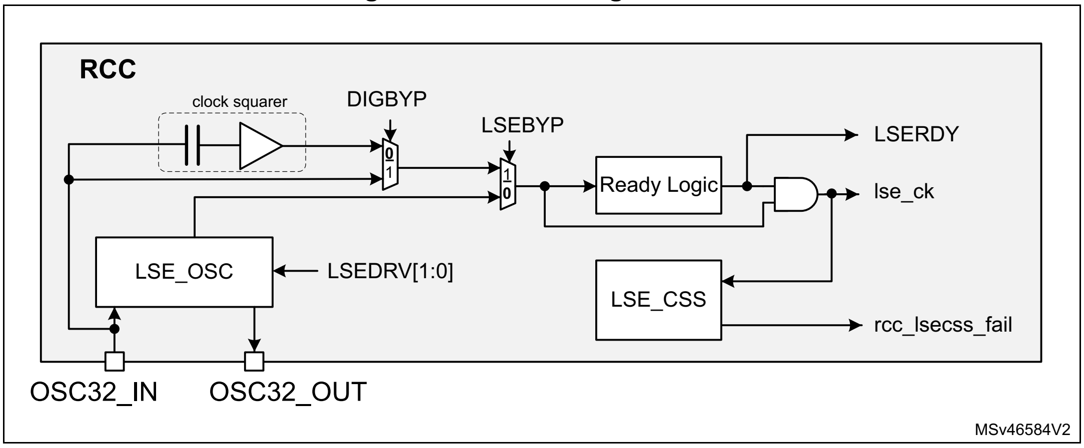
The LSE oscillator

The Low Speed External block can generate a clock from two possible sources:

- External crystal/ceramic resonator
- External user clock

The LSE function offers a LSERDY flag which indicates whether the LSE clock is available or not.

Figure 59. LSE clock generation



The LSE oscillator is switched on and off using the LSEON bit. The LSE remains enabled when the system enters in Stop, LP-Stop, LPLV-Stop, Standby or VBAT mode.

In addition, it is possible to drive the LSE clock to the MCO1 output and use it as a clock source for other application components.

External clock source (LSE bypass)

In this mode, an external clock source must be provided to OSC32_IN pin. The input clock can have a frequency up to 32.768 kHz, and be low-swing. This mode is selected by setting the LSEBYP and LSEON bits to ‘1’. In bypass mode, when the LSE is enabled (LSEON set to '1'), the LSERDY flag goes immediately to '1'.

The external clock signal can be digital or analog (square, sinus or triangle). An analog clock signal with a reduced amplitude, is supported thanks to an internal clock squarer. The input signal shall have a duty cycle close to 50%. The OSC32_OUT pin should be left HI-Z. Refer to the datasheet for additional information.

In order to select the external clock source, the application must follow the sequence hereafter:

- Set DBP bit of PWR_CR1 to ‘1’ in order to allow write accesses to RCC_BDCR register.
- Disable the LSE by writing to ‘0’ the LSEON bit,
- Check that the LSE is disabled by verifying that the bit LSERDY is set to ‘0’,
- Either set the bit DIGBYP to ‘1’ if the clock provided to OSC32_IN is full-swing digital signal, or set the bit DIGBYP to ‘0’ if the clock provided to OSC32_IN is low-swing signal,
- Set the bit LSEBYP to ‘1’,
- Enable again the LSE by writing to ‘1’ the bit LSEON,
- Check that the bit LSERDY is set to ‘1’, then the LSE is ready for use.
- Set DBP bit of PWR_CR1 to ‘0’ in order to write-protect accesses to RCC_BDCR register.

Note that those bits except DBP, are all located into the *RCC backup domain control register (RCC_BDCR)*.

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage of providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

In order to select the external crystal/ceramic resonator, the application must follow the sequence hereafter:

- Set DBP bit of PWR_CR1 to ‘1’ in order to allow write accesses to RCC_BDCR register.
- Disable the LSE by writing to ‘0’ the LSEON bit,
- Check that the LSE is disabled by verifying that the bit LSERDY is set to ‘0’,
- Set the bit LSEBYP to ‘0’,
- Configure LSEDRV[1:0] if needed,
- Enable again the LSE by writing to ‘1’ the bit LSEON,
- Check that the bit LSERDY is set to ‘1’, then the LSE is ready for use.
- Set DBP bit of PWR_CR1 to ‘0’ in order to write-protect accesses to RCC_BDCR register.

Note that those bits are all located into the *RCC backup domain control register (RCC_BDCR)*.

When the LSE is enabled (LSEON set to '1'), the LSERDY flag goes to '1' when 4096 cycles of LSE clock have been detected. This mechanism is also activated for an external clock source (LSEBYP = '1'). An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_MP_CIER)* or *RCC clock source interrupt enable register (RCC_MC_CIER)* when the LSE is ready.

Note: The LSE on MCO1 output is only available in Run and (LP-)Stop mode. In LPLV-Stop, Standby and VBAT mode LSE is not available from MCO1 output. When entering Standby or VBAT mode a spike may occur on MCO1.

The LSE offers a programmable driving capability (LSEDRV[1:0]) of the oscillator amplifier.It allows the application to adapt the oscillator to the characteristics of the external components. The driving capability shall be changed before enabling the LSE oscillator.

Warning: It is not allowed to change the driving capability when the LSE is enabled. The LSE behavior is not guaranteed in that case.

The LSI oscillator

The LSI acts as an low-power clock source that can be kept running when the System is in Stop, LP-Stop, LPLV-Stop or Standby mode for the independent watchdogs (IWDG) and Auto-Wakeup Unit (AWU). The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched on and off using the LSION bit, and the LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled.

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_MP_CIER)* or *RCC clock source interrupt enable register (RCC_MC_CIER)*.

In addition, it is possible to drive the LSI clock to the MCO1 output and use it as a clock source for other application components.

Note: Note that the LSI on MCO1 output is only available in Run and (LP-)Stop mode. In LPLV-Stop and Standby mode, the LSI is not available from MCO1 output. When entering Standby mode a spike may occur on MCO1.

Bits LSION and LSIRDY are located into the *RCC reset duration and LSI control register (RCC_RDLSICR)*.

10.4.3 Clock security system (CSS)

CSS on HSE

The RCC features a clock security system allowing the application to detect if the HSE clock stops toggling due to an unexpected reason. This could be caused for example, by a hardware failure or an attack. In the case failure detection the CSS will generate a system reset, and protect sensitive data of the BKPSRAM.

The clock security system can be activated by the application software via HSECSSON bit. The HSECSSON bit can be activated even when the HSEON is set to ‘0’. The CSS on HSE will be enabled by the hardware when the HSE is enabled and ready, and HSECSSON is set to ‘1’.

The CSS on HSE is disabled when the HSE is disabled (i.e. when the system is Stop or Standby).

It is not possible to clear the bit HSECSSON by software.

The bit HSECSSON will be cleared by the hardware when a system reset occurs or when the system enters in Standby mode. The bits HSEON and HSECSSON are located into the *RCC oscillator clock enable set register (RCC_OCENSETR)*.

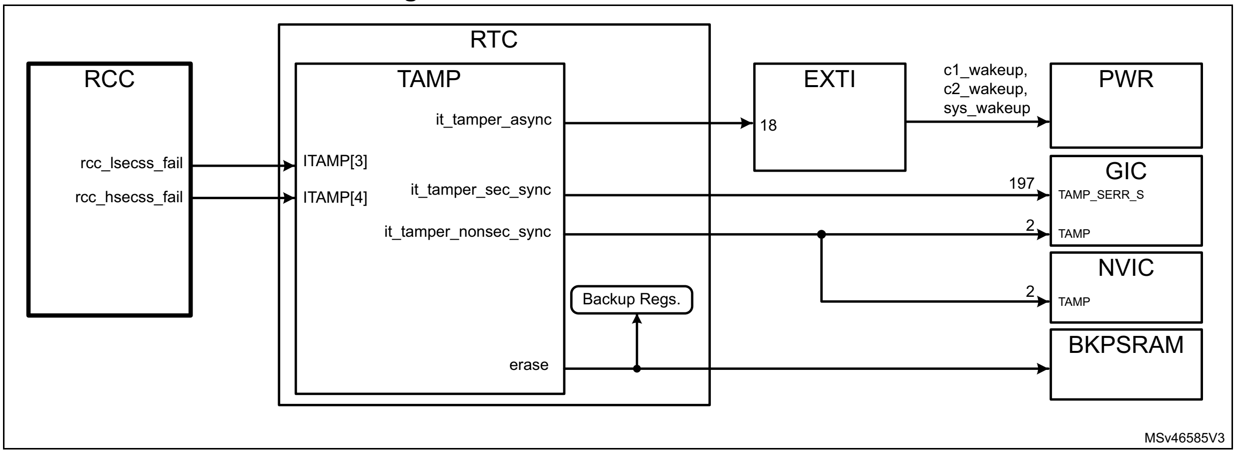
If a failure is detected on the HSE:

- The RCC generates an application reset (**app_rst** and **nreset**). The flag HCSSRSTF is set in order to allow the application to identify the reset root cause.
- A failure event is generated (**rcc_hsecss_fail**). This event is connected to the TAMP block, allowing the protection of backup registers and BKPSRAM.

Figure 60 shows the block involved in the HSE and LSE CSS function. Refer to the description of those blocks to get more details.

Note: Note that before selecting again the HSE clock, after an HSE failure, at least one the HCSSRSTF flags located into RCC BOOTROM reset status clear register (RCC_BR_RSTSCLRR) or RCC MCU reset status clear register (RCC_MC_RSTSCLRR) must be cleared. The BOOTROM performs this task by resetting the HCSSRSTF flag of the RCC_BR_RSTSCLRR *register, after an HSE failure. Then the BOOTROM reboots the circuit if the selected boot source does not require the HSE resource. Refer to BOOTROM documentation for details.*

Figure 60. LSE and HSE CSS function



CSS on LSE

The RCC features a clock security system allowing the application to detect if the LSE clock stops toggling due to an unexpected reason. This could be caused for example, by a hardware failure or an attack. In case of failure detection the CSS will generate a failure event, disable the clock provided to the RTC and protect sensitive data of the BKPSRAM.

The Clock Security System (CSS) on Low Speed External (LSE) oscillator can be activated by setting to ‘1’ the LSECSSON bit of the RCC_BDCR register.
This bit can be disabled only by hardware in the two following conditions:

- After a **nreset_vsw** reset (see *Figure 47*), or
- After a failure detection on LSE.

The activation of the CSS on LSE must be done in the following order:

1. Set the LSEON to ‘1’, and wait for LSERDY = ‘1’,
2. Select the LSE clock via the RTCSRC field,
3. Set the LSECSSON to ‘1’,

LSEON and RTCSRC are also located into the *RCC backup domain control register (RCC_BDCR)*.

The CSS on LSE is working in all system modes: Run, Stop, Standby and VBAT.

If a failure is detected on the LSE:

- The clock provided to the RTC is disabled immediately by the hardware, by setting the RTCSRC field to ‘0’.
- The values of LSEON and LSECSSON are not changed by the hardware.
- A failure event is generated (**rcc_lsecss_fail**). This event is connected to the TAMP block, allowing to wake up from Standby, and allowing the protection of backup registers and BKPSRAM.
- An interrupt flag (LSECSSF) is activated in order to generate an interrupt to the MCU or MPU if needed.

The CSS on LSE offers two flags in order to detect a failure: the flag LSECSSD located into the RCC_BDCR register, and the LSECSSF located into the *RCC clock source interrupt flag register (RCC_MP_CIFR)*.

In case of failure, the software has to do the following:

- Disable the CSS function (this step is mandatory)
  – Set the LSECSSON bit to ‘0’
  – Set the LSEON bit to ‘0’, in order to disable the LSE
  – Clear the LSECSSF bit, if interrupt was enabled for this event
- Change the clock source for the RTC if needed:
  – Set the RTCEN bit to ‘0’ to disable the RTC clock
  – Enable the new clock source for the RTC
  – Select the proper clock source via RTCSRC
  – Set the RTCEN bit to ‘1’ to enable the RTC clock

For sure, the software can take any other required actions to secure the application.

Figure 60 shows the block involved in the HSE and LSE CSS function.

Note: The clock security feature must not be re-enabled.
Before re-using the CSS function for LSE, a backup domain reset must be performed for example via the VSWRST bit in RCC backup domain control register (RCC_BDCR).

10.4.4 Clock output generation (MCO1 and MCO2)

Two micro-controller clock output (MCO) pins are available: MCO1 and MCO2. For each output, it is possible to select a clock source.The selected clock can be divided thanks to configurable prescaler. Refer to *Figure 54* for additional information on signal selection.

The MCO1 and MCO2 outputs are controlled via MCO1DIV[3:0], MCO1SEL[2:0], MCO2DIV[2:0] and MCO2SEL[2:0] located into the *RCC MCO1 configuration register (RCC_MCO1CFGR)* and *RCC MCO2 configuration register (RCC_MCO2CFGR)*.

For the different MCO pins, the corresponding GPIO port has to be programmed in alternate function mode.

The dividers MCODIV1 and MCODIV2, provide a clock with a duty cycle of 50% for divisions with even values. More generally the duty cycle is given by the following formula:

$$DC(\%) = \frac{FLOOR\left(\frac{MCODIVx + 1}{2}\right)}{(MCODIVx + 1)} \times 100$$

For MCODIVx = 0, the duty cycle is also 50%.

Note that the MCO1 and MCO2 outputs are only available in Run and (LP-)Stop modes.

Caution: The clock provided to the MCOs outputs must not exceed the maximum PAD speed, Refer to the product datasheet for information about the supported PAD speed.

10.4.5 PLLs description

The RCC features four PLLs:

Two PLLs having a VCO frequency of up to 1.6 GHz (PLL_1600):

- A PLL (PLL1) exclusively used to provide the clock to the Cortex-A7 MPU Sub-System (CA7-SS).
- A PLL (PLL2) used to the clock generation for the AXI-SS. Some outputs of this PLL are also used to generate a kernel clock to peripherals.

Two PLLs having a VCO frequency of up to 800 MHz (PLL_800):

- A PLL (PLL3) used to provide the clock to the MCU and its interconnect.
- A PLL (PLL4) Dedicated to the generation of kernel clocks to peripherals.

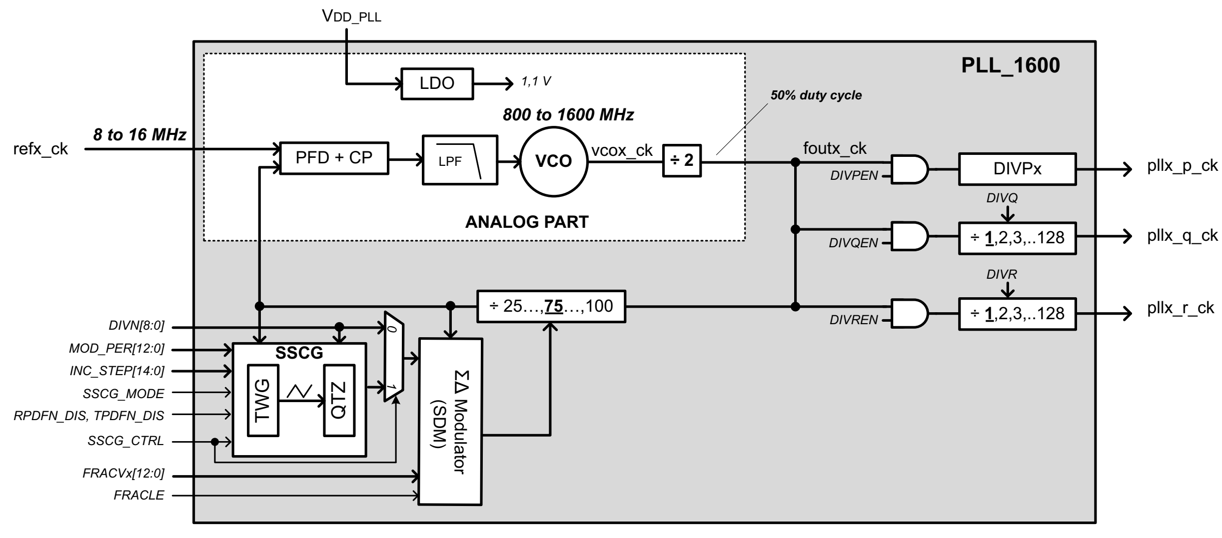
PLL_1600 main features:

- VCO range between 800 and 1600 MHz
- Input frequency range: 8 to 16 MHz
- Three working modes
  - Fractional mode, using 13-bit sigma-delta modulator. The sigma-delta modulator can be updated on-the-fly, without generating frequency overshoots on PLLs outputs.
  - Integer mode
  - Spread spectrum mode in order to reduce EMI
- Up to 3 outputs:
  - 3 outputs with post-dividers

PLL_800 main features:

- VCO range between 400 and 800 MHz
- Input frequency range: 4 to 16 MHz
- Three working modes
  - Fractional mode, using 13-bit sigma-delta modulator. The sigma-delta modulator can be updated on-the-fly, without generating frequency overshoots on PLLs outputs.
  - Integer mode
- Up to 3 outputs:
  - 3 outputs with post-dividers

Figure 61. PLL1 and PLL2 block diagram

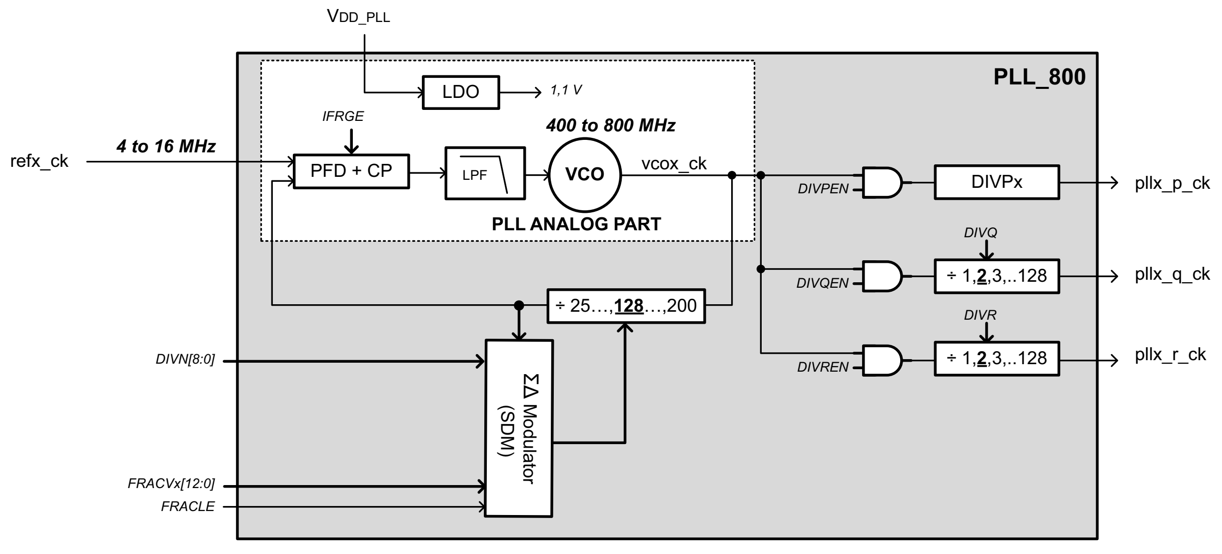


(1) It is recommended to use the values 1,2,4,6...128 for DIVP of PLL1 and PLL2

The PLL1 and PLL2 analog parts provide a dedicated outputs delivering a clean clock signal (**foutx_ck**) with a duty cycle of 50%. The **foutx_ck** clock has a frequency of F_{VCO} / 2, and is available on **pllx_p_ck, pllx_q_ck** and, **pllx_r_ck** outputs when the post-dividers are bypassed.

The MPU and DDRC are very sensitive to clock jitter and duty cycle degradation, the application shall use directly **foutx_ck** when possible.

Figure 62. PLL3 and PLL4 block diagram



(1) It is recommended to use the values 1,2,4,6...128 for DIVP of PLL3

The PLLs are controlled via RCC_PLLxCR, RCC_PLLxCFGR1, RCC_PLLxCFGR2 and RCC_PLLxFRACR registers.

The user must provide to each PLL, a clean reference clock (**refx_ck**) in the accepted input frequency range, which is:

- 8 to 16 MHz for the PLL1 and PLL2
- 4 to 16 MHz for the PLL3 and PLL4

The application has to program properly the DIVM1, DIVM2, DIVM3 and DIVM4 dividers in order to match these conditions.

In addition, the IFRGE field must be set according to the reference input frequency: it will guarantee an optimal performance of the PLL3 and PLL4.

The loop divider DIVN has to be programmed in order to get the expected frequency at VCO output. The VCO output range must be respected.

The PLLs can be enabled by setting the corresponding PLLON bit to ‘1’. The PLLxRDY bits inform the user that the PLL is ready (i.e. locked).

Warning: For the DIVP outputs of PLL1 and PLL2, only the bypass (division by 1) and even divisions are allowed.
For the DIVP output of PLL3, only the even divisions are allowed.
If those conditions are not respected the behavior of the circuit is not guaranteed.

More generally, in order to get PLLs output frequencies with a duty cycle close to 50%, DIVP, DIVQ, DIVR must select an even division.

PLL programming recommendations

This section is providing a list of recommendations to follow in order to guarantee a good usage of the PLLs. Note that programming examples are given in *Section 10.6.2: PLL programming*.

- Before enabling the PLLs, the user must ensure that the reference frequency (**refx_ck**) provided to the PLL is in the good range.
- When the PLLs are enabled, it is possible to change the values of a post-dividers (DIVP, DIVQ or DIVR) without disabling the corresponding PLL.
  Refer to *Section : Reprogramming post-dividers* for details.
- When the PLLs are enabled, changing the value of DIVMx, DIVNx, IFRGE conducts to an invalid output frequency, with the risk of crashing the system. There is no hardware protection, it is up to the application to avoid this situation.
- When one or several PLL outputs are not used, the application shall set the corresponding enable bit DIVyEN to ‘0’.
- In order to ensure the good behavior of the PLL, the bits DIVPEN, DIVQEN and DIVREN must be set to '0' before disabling the corresponding PLL. In the same way, the bits DIVPEN, DIVQEN and DIVREN can be set to '1' only if the correspond bit PLLON is already set to '1' and the PLL is ready.

*Note: The **refx_ck** clock is provided to the PLLx when the corresponding PLLON bit is set to ‘1’.*

The PLLs can work in 3 different modes:

- In integer mode
- In fractional mode
- In spread spectrum mode (only for PLL1 and PLL2)

Using the PLLs in integer mode

The PLL is working in integer mode when the sigma-delta modulator (SDM) is loaded with a '0', and the bit SSCG_CTRL = '0' where available.

In order to load '0' into the SDM, the user has to perform the following sequence:

- Write FRACV to '0', and FRACLE to '0'
- Write FRACLE to '1'

The bits FRACV[12:0] and FRACLE are located in the RCC PLLx Fractional Register (RCC_PLLxFRACR) with x = 1, 2, 3 or 4.

For the PLL1 and PLL2, the VCO frequency (F_{VCO}) is given by the following expression:

$$F_{VCO} = F_{ref\_ck} \times 2 \times (DIVN + 1)$$

And the frequency at the output of the post-dividers is given by:

$$F_{pll\_y\_ck} = \frac{F_{VCO}}{2 \times (DIVy + 1)}$$

with y = P, Q or R

For the PLL3 and PLL4, the VCO frequency (F_{VCO}) is given by the following expression:

$$F_{VCO} = F_{ref\_ck} \times (DIVN + 1)$$

And the frequency at the output of the post-dividers is given by

with y = P, Q or R

$$F_{pll\_y\_ck} = \frac{F_{VCO}}{DIVy + 1}$$

Using the PLLs in fractional mode

The fractional mode is activated when the value loaded into the SDM is different from 0, and the bit SSCG_CTRL = '0' where available.

The SDM value can be updated at anytime by the application by executing the following sequence:

- Write into FRACV the new fractional value, and FRACLE to '0'
- Write FRACLE to '1'. The new FRACV value is loaded into the SDM when FRACLE bit goes from 0 to '1'.

The sigma delta modulator is designed in order to minimize the jitter impact while allowing very small frequency steps.

When the PLL is used in fractional mode, the DIVN divider must be initialized before enabling the PLL, but it is then possible to change the value of FRACV on-the-fly without disturbing the PLL output.

This feature can be used either to generate a specific frequency, with a good accuracy from any crystal value, or to perform a fine tuning of the frequency on-the-fly.

For the PLL1 and PLL2, the VCO frequency is given by the following expression:

$$F_{VCO} = 2 \times F_{ref\_ck} \times \left((DIVN + 1) + \frac{FRACV}{2^{13}} \right)$$

$F_{pll\_x\_ck}$ can be computed using formula of *Section : Using the PLLs in integer mode*.

For the PLL3 and PLL4, the VCO frequency is given by the following expression:

$$F_{VCO} = F_{ref\_ck} \times \left((DIVN + 1) + \frac{FRACV}{2^{13}} \right)$$

$F_{pll\_x\_ck}$ can be computed using formula of *Section : Using the PLLs in integer mode*.

Using the PLLs in spread spectrum mode

The spread spectrum mode is activated when the SDM loaded with 0, and the bit SSCG_CTRL = ‘1’. This feature is available on PLL1 and PLL2.

The spread spectrum technique consist of modulating the VCO frequency with a low-frequency signal (in our case a triangular signal), in order to spread the clock energy into a wider frequency band. As a consequence, the amount of emitted EMI peaks will be reduced.

The parameters of the spread spectrum modulation are adjusted via the following fields:

- MOD_PER[12:0]: used to adjust the modulation frequency

Note: Note that the modulation frequency is generally in the range of 20 to 50 kHz.

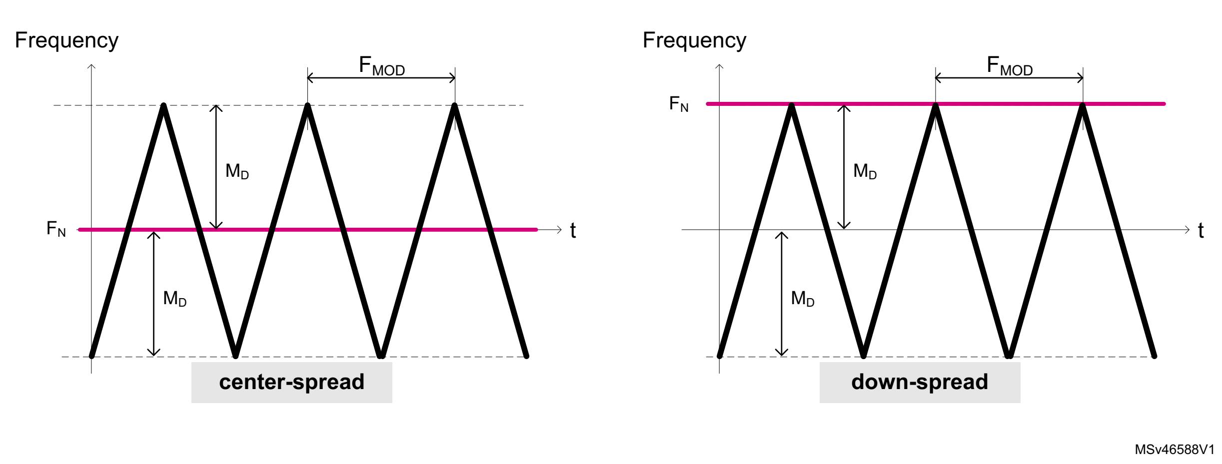
- INC_STEP[14:0]: used to adjust the modulation depth (or modulation index)
- SSCG_MODE: used to define if the modulation is centered around the VCO frequency (center-spread) or if it is lowered with respect to VCO frequency (down-spread).

The bits MOD_PER[12:0], INC_STEP[14:0] and SSCG_MODE are located in RCC PLLx Clock Spreading Generator Register (RCC_PLLxCSGR) with x=1 or 2.

Figure 63 shows the signal modulating the nominal frequency (F_N), when SSCG_MODE = 0 (center-spread) and SSCG_MODE = ‘1’ (down-spread). The nominal frequency is the frequency output by the PLL in integer mode, when no clock spreading is applied.

Setting the SSCG_MODE bit to ‘1’ (down-spread) guaranties that the PLL output frequency will not exceed the programmed frequency value when the SSCG will be enabled.

Figure 63. Spread spectrum modulation



The peak modulation depth (in percentage) is given by the following formula:

$$M_D\,(\%) = \frac{MOD\_PER \times INC\_STEP \times 100 \times 5}{(2^{15} - 1) \times (DIVN + 1)}$$

Note that MOD_PER x INC_STEP shall not exceed (2^{15}-1).

The modulation frequency (F_{mod}) is given by:

$$F_{mod} = \frac{F_{ck\_ref}}{4 \times MOD\_PER}$$

In order to use the spread spectrum feature, the user has to do the following:

- Program the PLL according to *Section : Using the PLLs in integer mode* in order to adjust the nominal frequency (F_N) according to the targeted by the application.
- Compute the MOD_PER value according to the wanted modulation frequency (F_{mod}):

$$MOD\_PER = ROUND\left(\frac{F_{ck\_ref}}{4 \times F_{mod}}\right)$$

- Compute the INC_STEP value according to the wanted modulation depth (M_D):

$$INC\_STEP = ROUND\left(\frac{(2^{15} - 1) \times M_D \times (DIVN + 1)}{100 \times 5 \times MOD\_PER}\right)$$

  Check that MOD_PER x INC_STEP does not exceed (2^{15}-1).

- Then DIVP,Q or R can be adjusted, and the bit SSCG_CTRL can be set to ‘1’, and the PLL enabled.

The user can check F_{MIN},F_{MAX} and F_C as follow:

- If SSCG_MODE = ‘0’ (centered-spread)

  F_C is given by the formula of the *Section : Using the PLLs in integer mode*

  $F_{MIN} = F_C \times (1 - M_D/100)$

  $F_{MAX} = F_C \times (1 + M_D/100)$
- If SSCG_MODE = ‘1’ (down-spread)

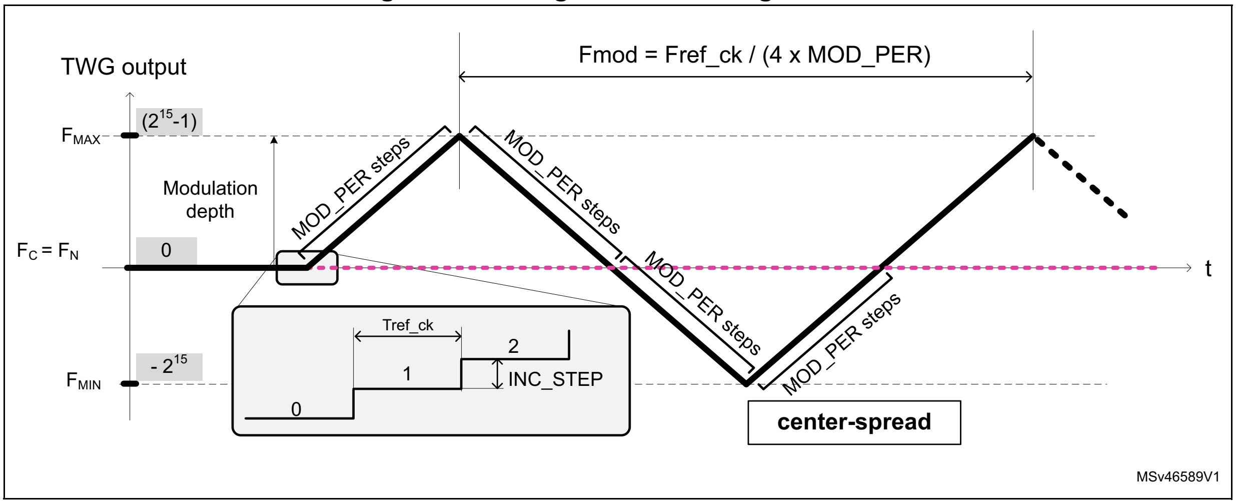
  F_{MAX} is given by the formula of the *Section : Using the PLLs in integer mode*

  $F_{MIN} = F_{MAX} \times (1 - 2 \times M_D/100)$

  $F_C = F_{MAX} \times (1 - M_D/100)$

Figure 64 shows the digital signal generated by Triangular Waveform Generator (TWG block), and the way of MOD_PER and INC_STEP are changing the triangular waveform.

Figure 64. Triangular waveform generator



Refer to *Section : Reprogramming post-dividers* and *Section : PLL initialization procedure* for additional information on the PLL programming.

10.4.6 Sub-system clocks

The RCC handles three sub-system clocks: **mpuss_ck**, **axiss_ck** and **mcuss_ck**.

Sub-system clock selection

After a system reset, the HSI is selected as the main clock for the complete system. So the HSI is used to provide the clocks:

- **mpuss_ck**,
- **axiss_ck**,
- mcuss_ck

After a system reset, all PLLs are switched off, as well as CSI and HSE.

When the system is running, the user has the possibility to select the clock to be used for the MPU (**mpuss_ck**), for the interconnect (**axiss_ck**), and for the MCU (**mcuss_ck**).

For **mpuss_ck**, the application can select between the following sources:

- HSI
- HSE or
- pll1_p_ck
- **pll1_p_ck** / MPUDIV

For **axiss_ck**, the application can select between the following sources:

- HSI
- HSE or
- pll2_p_ck

For **mcuss_ck**, the application can select between the following sources:

- HSI
- HSE
- CSI or,
- pll3_p_ck

The clock source for **mpuss_ck** is controlled by MPUSRC located into the *RCC MPU clock selection register (RCC_MPCKSELR)*, the clock source for **axiss_ck** is controlled by AXISSRC located into the *RCC AXI sub-system clock selection register (RCC_ASSCKSELR)*, and the clock source for **mcuss_ck** is controlled by MCUSSRC located into the *RCC MCU sub-system clock selection register (RCC_MSSCKSELR)*.

Dynamic clock switches

The clock selectors shown with a square D on *Figure 65* allows to reconfigure the selected input on the fly without spurs or timing violation. This allows dynamic clock switching.

As a consequence, the switching from the original to the new input can only be performed if a clock is present on both inputs. If it is not the case, no clock will be provided. In order to recover from this situation the user has to provide a valid clock to both inputs.

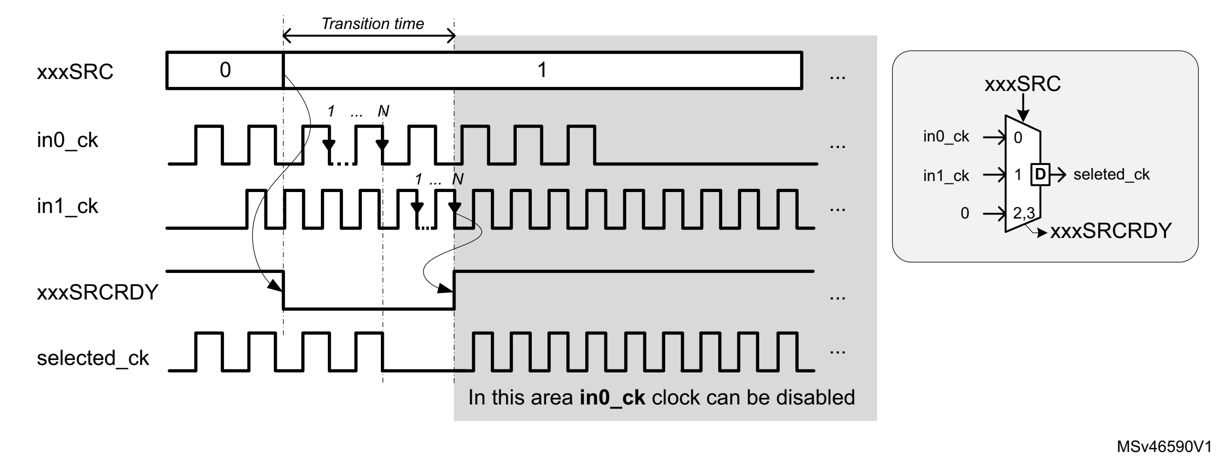
During the transition from one input to the other, the clock output will be gated, in the worst case, during 3 clock cycles of the previously selected clock and 3 clock cycles of the new selected clock. As shown in *Figure 65*, both input clocks shall be present during transition time.

Note that some dynamic switches such as the ones controlled by PLL12SRC, PLL3SRC, PLL4SRC, MPUSRC, AXISSRC and MCUSSRC provide a ready signal (xxxSRCRDY in *Figure 65*). This ready signal goes immediately to ‘0’, when the clock source is changed (xxxSRC), and goes to ‘1’ when the switch is able to provide the new selected clock. This signal can be used to check that the switch transition has been properly done.

When those switches are placed in position where no input clock is selected (position 2,3 in the example of *Figure 65*), no clock will be generated to their outputs, and the xxxSRCRDY flag will be set to ‘0’.

*Note: Note however that the xxxSRCRDY flag cannot be used to detect if a clock is generated at the switch output. For example, if the switch is already selecting the **in0_ck**, and the xxxSRCRDY flag is set to ‘1’, if the **in0_ck** is disabled, the xxxSRCRDY flag will not go to ‘0’, but stays to ‘1’ because last switch transition has been properly performed. For sure if the application changes the xxxSRC value, if the **in0_ck** is disabled the switch will not be able to selected the new input, and the xxxSRCRDY flag will remain to ‘0’.*

Figure 65. Dynamic clock switching



Hardware protection for switch transition

The switching from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switching occurs when the clock source is ready.

The status bits in the *RCC oscillator clock ready register (RCC_OCRDYR)* indicate which clock(s) is (are) ready.

Sub-system clock generation

Figure 66 shows a simplified view of the clock distribution for the CPUs and busses. All the dividers presented in the block diagram can be changed on-the-fly without generating timing violations. This feature allows the user to adapt in a very simple way, the busses frequencies to the application needs, and thus optimize the power consumption.

The MPUDIV divider located into *RCC MPU clock divider register (RCC_MPCKDIVR)* can be used to adjust dynamically the clock for the MPU.

The AXIDIV divider located into *RCC AXI clock divider register (RCC_AXIDIVR)* can be used to adjust dynamically the clock for the AXI matrix. The value of this divider also impacts the clock frequency of AHB5, AHB6, APB4 and APB5.

The APB4DIV divider located into *RCC APB4 clock divider register (RCC_APB4DIVR)* can be used to adjust dynamically the clock for the APB4 bridge.

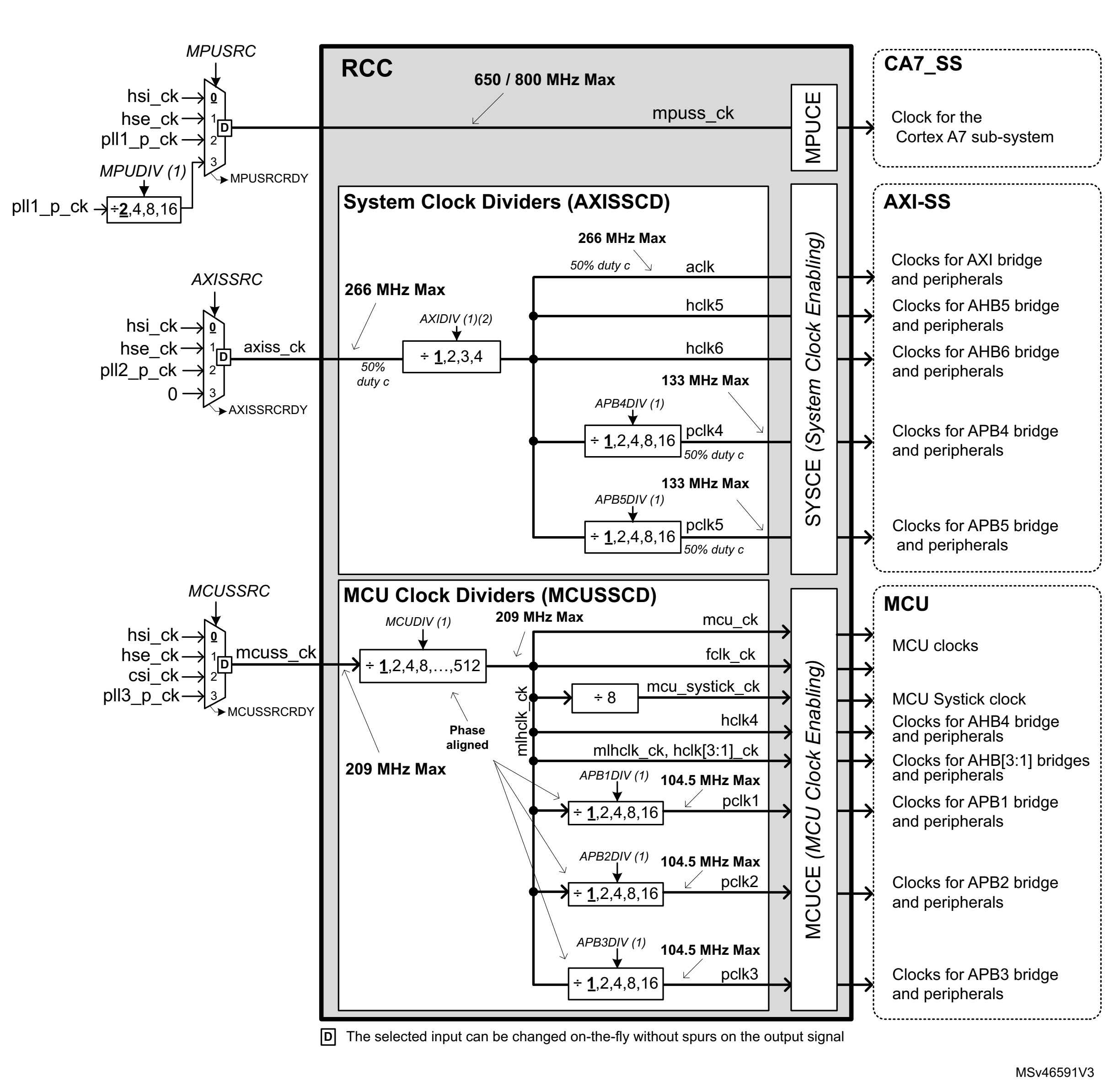
In the same way, the APB5DIV divider located into *RCC APB5 clock divider register (RCC_APB5DIVR)*, can be used to adjust dynamically the clock for the APB5 bridge.

In the MCU clock tree, the MCUDIV divider located into *RCC MCU clock divider register (RCC_MCUDIVR)*, can be used to adjust dynamically the clock for the MCU. The value of this divider also impacts the clock frequency of AHB bridges, APB1, APB2 and APB3.

The APB1DIV divider located into *RCC APB1 clock divider register (RCC_APB1DIVR)*, can be used to adjust dynamically the clock for the APB1 bridge. Similar implementation is available for APB2 and APB3.

Note that dividers having the possibility to be updated on-the-fly provides a ready bit in order to inform the application on the moment when the new division ratio is taken into account.

Figure 66. Core and busses clock generation



1. Can be changed on the fly.
2. Keeps a duty-cycle of 50% even for division by 3.
3. **X** represents the selected MUX input after a system reset.

10.4.7 Clock generation in Stop and Standby modes

The product supports three kinds of system Stop modes:

- Normal system stop mode, named Stop
- Low-power system stop mode, named LP-Stop
- Low-power, low-voltage system stop mode, named LPLV-Stop

The main differences between those stop modes is the handling of the V_{DDCORE} supply voltage:

- In Stop mode, the V_{DDCORE} supply voltage remains the same as system Run mode, with the same power capability,
- In LP-Stop mode, the V_{DDCORE} supply voltage remains the same as system Run mode, but with a limited power capability. This is usually achieved thanks to an external PMIC having 2 power modes: a high power mode (HP) for nominal cases (Run and Stop) and a low-power mode (LP) limiting the power capability for LP-Stop and LPLV-Stop modes.
- In LPLV-Stop mode, the V_{DDCORE} supply voltage is reduced, and the power capability is reduced as well,

The amount of peripherals able to wake-up the system from Stop, depends on the Stop mode activated. For example, Stop or LP-Stop modes offer a larger choice of peripherals able to wake-up the system than the LPLV-Stop mode. Please refer *Section Table 34.: Functionalities depending on system operating mode* in the PWR.

Going to Stop, LP-Stop or LPLV-Stop mode

The system can go to Stop, LP-Stop or LPLV-Stop mode, when the MPU is in CStop or CStandby, and the MCU in CStop. Refer to *Section 9.5.1: PWR operating modes*.

Note as well that before going to CStop, each processor has to mask all the interrupts except the RCC wakeup interrupts in order to ensure a safe Stop sequence entry and exit. The RCC provides two wakeup interrupts: **rcc_mpu_wkup_it** and **rcc_mcu_wkup_it** which can be enabled via their corresponding WKUPIE bit. Refer to *Section 10.5: RCC interrupts* for additional information.

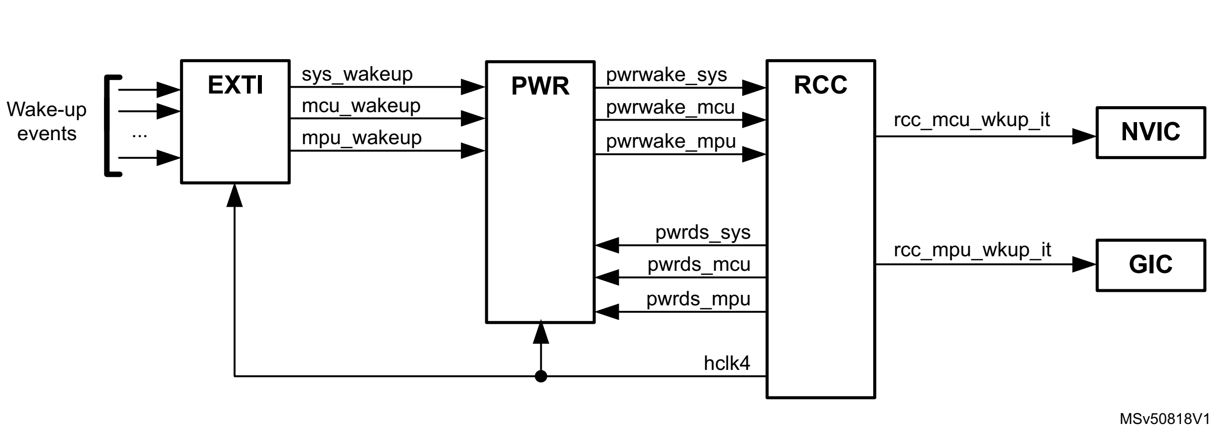
Before switching off all the clocks, the RCC waits that the DDRC enters into self-refresh mode. As a consequence, the effective entry into Stop, LP-Stop or LPLV-Stop mode, may be delayed until the DDRC allows it. Then the RCC stops all the clocks (sub-system and kernel clocks) and the following clock sources are disabled as well:

- CSI, if CSIKERON = '0',
- HSI, if HSIKERON = '0',
- HSE, if HSEKERON = '0',
- PLL1, PLL2, PLL3 and PLL4.

Note: In LPLV-Stop mode, CSI, HSI and HSE cannot be used.

Note: LSE and LSI oscillators remain active in Stop, LP-Stop and LPLV-Stop mode if enabled.

Figure 67. Key signals controlling low-power modes



In addition, the RCC informs the PWR block that the MPU and/or MCU sub-system are in CStop via signals **pwrds_mpu**, **pwrds_mcu** and **pwrds_sys** (see *Figure 46*).

- The **pwrds_mpu** is used to indicate to the PWR that the MPU is in CStop or CStandby.
- The **pwrds_mcu** is used to indicate to the PWR that the MCU is in CStop.
- The **pwrds_sys** is used to indicate to the PWR that the system is in Stop, LP-Stop or LPLV-Stop, and no longer clocked.

Note that when only one processor goes to CStop, the PLLs and oscillators generating **mpuss_ck**, **axiss_ck**, **mcuss_ck** and kernel clocks are kept activated.

During Stop, LP-Stop or LPLV-Stop mode

As explained in the previous section, in system Stop, LP-Stop or LPLV-Stop mode, all clocks, PLLs and oscillators are disabled, except LSE and LSI which are kept in their current states. However there are two specific cases in Stop and LP-Stop mode where the HSE, HSI or CSI can be enabled:

- When a dedicated peripheral is requesting a kernel clock:
  In that case the peripheral will receive the HSE, HSI or CSI according to the kernel clock source selected for this peripheral (via PERxSRC).
- When the bits HSEKERON, HSIKERON or CSIKERON are set:
  In that case the HSE, HSI or CSI are kept running during (LP-)Stop mode, but the clock is not provided to the peripherals. In that way, the clock will be available immediately when the system exits from (LP-)Stop or when a peripheral is requesting the kernel clock. Refer to *Table 54* for details.

The HSIKERON, HSEKERON and CSIKERON bits are located into the *RCC oscillator clock enable set register (RCC_OCENSETR)* and *RCC oscillator clock enable clear register (RCC_OCENCLRR)*. *Table 54* gives details on the behavior of those bits.

Note as well that the HSE, HSI and CSI outputs provide two clock paths (see *Figure 54*):

- One path for the processors and interconnects clocks (**hse_ck**, **hsi_ck** or **csi_ck**),
- One path for the peripheral kernel clocks (**hse_ker_ck**, **hsi_ker_ck** or **csi_ker_ck**).

In system (LP-)Stop mode, when a peripheral requests the kernel clock, only the path providing the peripheral kernel clock is activated, the processors and interconnects clock paths remain inactive.

Table 54. HSI, CSI and HSE states versus system modes(1)

System modes	HSION (CSION) (HSEON)	HSIKERON (CSIKERON) (HSEKERON)	Kernel clock request (2)	-	HSI State (CSI State) (HSE State)	hsi_ck (csi_ck) (hse_ck)	hsi_ker_ck (csi_ker_ck) (hse_ker_ck)
Run	0	X	1	→	ON	Gated	ON
			0	→	OFF		Gated
	1		X	→	ON	ON	ON
Stop or LP-Stop	X	X	1	→	ON (3)	Gated	ON
		0	0	→	OFF		Gated
		1	0	→	ON		ON (4)
Standby or LPLV-Stop	X	X	X	→	OFF	Gated	Gated

1. 'ON' state highlighted in gray.
2. Kernel clock request is equal '1' when the peripheral is requesting a kernel clock AND its corresponding PERxSRC is selecting the HSE, HSI or CSI oscillator, AND if its PERxEN and PERxLPEN are set to '1'. Refer to *Table 63: Peripheral clock enable details for MPU (MCU)* for details.
3. Only the oscillator selected by the PERxSRC of the peripheral requesting the kernel clock is enabled.
4. The kernel clock is provided to the PCKE block, and gated as long as the peripheral does not generate a kernel clock request.

Leaving Stop, LP-Stop or LPLV-Stop mode

The system can leave Stop, LP-Stop or LPLV-Stop mode thanks to events generated by peripherals able to work in the selected Stop mode. The exit from system Stop is generally done in the following way (see also *Figure 67*):

- A peripheral sends a wakeup interrupt to the EXTI block.
- The EXTI block generates an event to the PWR block (**sys_wakeup**).
- The PWR block signals to the external PMIC that the main power should be restored (through PWR_LP signal or optionally PWR_ON signal).
- If the PMIC was in LPLV-Stop, the PWR block waits for a nominal V_{DDCORE} voltage.
- When V_{DDCORE} reached the nominal value, then the PWR asserts the **pwrwake_sys** for the RCC.
- Thanks to this signal, the RCC provides the **hclk4** clock to the EXTI and PWR blocks and starts the clock restore sequence.
- When the **hclk4** is available, the EXTI asserts the **mpu_wakeup**, or **mcu_wakeup** signals, indicating to the PWR block which processor shall be activated.
- The PWR then asserts the **pwrwake_mpu**, or **pwrwake_mcu** signals for the RCC according to the processor to be activated.
- On the RCC side:
  - If the signal **pwrwake_mpu** is activated, the RCC asserts the **rcc_mpu_wkup_it**, and provides a clock to the MPU and AXI sub-system when those clocks are available.
  - If the signal **pwrwake_mcu** is activated, the RCC asserts the **rcc_mcu_wkup_it**, and provides the **ck_hsi** to the MCU when available.
- Then the targeted processor exits from CStop, and re-enables the interrupts of peripherals activated during Stop mode. In that way, the peripheral which has generated the wakeup event is able to generate an interrupt to the sub-system's processor.
- The RCC acknowledges the **pwrwake_mpu**, or **pwrwake_mcu** requests by de-asserting the **pwrds_mpu** or **pwrds_mcu** signals.

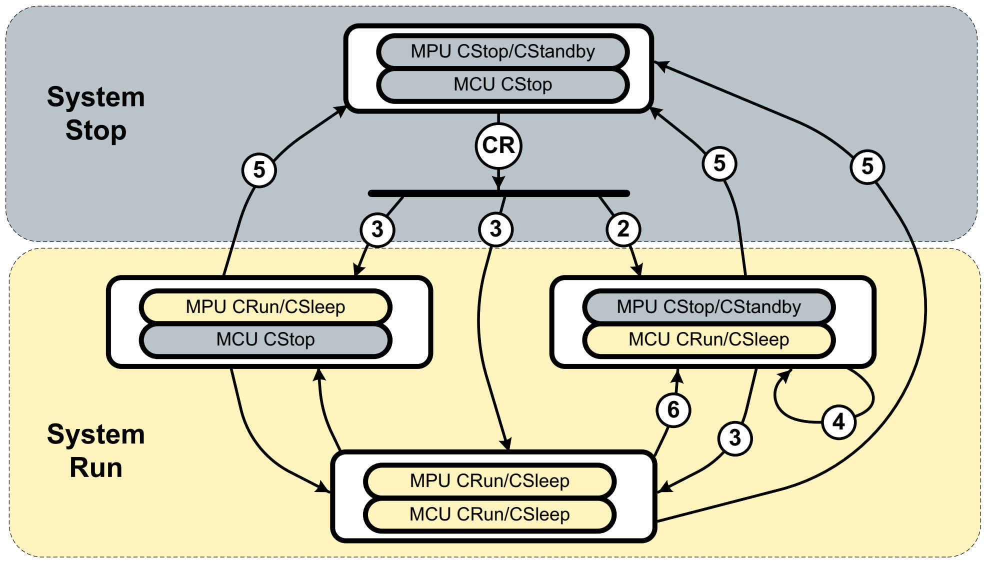
Caution: The RCC integrates a state machine able to restore the clock configuration which was used before going to one of the Stop modes. The PLL3 and PLL4 are not handled by this mechanism.

When exiting from one of the system Stop modes, the MCU always uses the **hsi_ck** as sub-system clock and the PLL3 and PLL4 are disabled. As a consequence, after a system Stop, the following bits are set to their reset value:

- HSION = '1', MCUDIV = '0', MCUSSRC = '0',
- PLL3ON = '0', DIVPEN, DIVQEN and DIVREN of PLL3 are set to '0'
- PLL4ON = '0', DIVPEN, DIVQEN and DIVREN of PLL4 are set to '0'

The RCC executes the clock restore sequence every time the system exits from Stop, LP-Stop or LPLV-Stop as shown in *Figure 68*. When the system is in Run, all oscillators and PLLs requested by the application are activated, but the sub-system clocks are only provided according to MCU and MPU modes.

Figure 68. Clock restore activation versus system mode



CR Clock restore sequence triggered by **pwrwake_sys** event. **hsi_ck** is enabled in order to generate **hclk4**. PLL1 and PLL2 are re-enabled if their respective PLLxON bit was set to ‘1’.

2 Enabling of **mcuss_ck** clock (**pwrwake_mcu**)

3 Enabling of **mpuss_ck, mcuss_ck** and **axiss_ck** clocks, if not already enabled (**pwrwake_mpu**)

4 Clock enabling of **axiss_ck**

5 Disablling of all sub-system clocks, all PLLs, and oscillators if allowed

6 Disabling of **mpuss_ck** and **axiss_ck** if not needed by MCU

Run, Crun, or Csleep — Stop, Cstop, or CStandby

MSv46592V1

Handling the PWRLP_TEMPO

The PWRLP_TEMPO is used to wait automatically that the external PMIC switches from Low-power (LP) mode to High-power (HP), before enabling power-consuming parts. The application can control the delay value via *RCC PWR_LP delay control register (RCC_PWRLPDLYCR)*. The PWRLP_TEMPO is using the **hsi_ck** clock as reference.

In order to allow a fast restart time when the MCU wakes up from system (LP-)Stop or LPLV-Stop, it is possible for the activation of the **mcuss_ck** clock to bypass the PWRLP_TEMPO delay. This is only possible when the power consumption of the MCU is low enough to be supported by the PMIC LP mode. This function is controlled by the MCTMPSKP bit of the RCC_PWRLPDLYCR register. Note that in this case, the PMIC still transitions from LP to HP mode.

Refer to *Section 10.6.5: Clock restore sequence examples* for additional information.

Warning: **When the bit MCTMPSKP is set to ‘1’, the application must ensure that the MCU did not allocate peripherals located into the AXI sub-system (i.e. on AXIM, AHB5, AHB6, APB4 or APB5). If this rule is not respected, a system reset may be generated.**

The clock restore sequence description

Every time the system exits from one of the Stop modes (**pwrwake_sys**), the RCC re-enables the requested oscillators and PLLs in order to restore the **mpuss_ck, axiss_ck** and **mcuss_ck** clocks. The sub-system clock settings are defined by the registers listed on *Table 55*.

The registers listed on *Table 55* cannot be modified during the clock restore sequence.

If the system exits from one of the Stop modes due to a wakeup event for the MPU (**pwrwake_mpu**), then the **mpuss_ck, mcuss_ck** and **axiss_ck** clocks will be re-enabled simultaneously only when both **mpuss_ck**, **mcuss_ck** and **axiss_ck** are ready.

If the system exits from one of the Stop modes due to a wakeup event for the MCU (**pwrwake_mcu**), then the **mcuss_ck,** and **axiss_ck** (if needed for MCU) clocks will be re-enabled simultaneously only when both **mcuss_ck** and **axiss_ck** are ready. The clock **mpuss_ck** and **axiss_ck** (if not needed) will remain gated even if their respective PLLs are enabled. The **axiss_ck** clock will be enabled if the MCU allocated at least one peripheral into the AXI sub-system. Note that the **mcuss_ck** is always connected to **hsi_ck** when the system exits from Stop.

More precisely, the clock restore sequence will be split into the following main steps (see also *Figure 69* and *Figure 70*):

- Set the bit CKREST = ‘1’ and AXICKRDY = ‘0’ in order to inform that a clock restore sequence will be initiated. The registers listed on *Table 55* cannot be modified during the clock restore sequence.
- Enable the oscillators which were enabled before going to one of the system Stop modes.
- When the HSI is ready, the RCC can trigger the PWRLP_TEMPO (according to PWRLP_DLY value) in order to allow a delay to the external PMIC to switch in HP mode, before doing power-consuming operations on V_{DDCORE}.
- When the oscillator used by PLL1 and PLL2 is ready, the PLLs analog parts can be powered. As the PLLs analog parts are not powered by V_{DDCORE}, it is possible to enable the VCOs, with all outputs (P, Q, and R dividers) disabled. There is no need to wait that PWRLP_TEMPO elapsed.
- Before enabling the PLL outputs, the RCC shall wait until the PWRLP_TEMPO elapsed, and the PLLs are ready. The process generating the **mcuss_ck** clock may also need to wait until the PWRLP_TEMPO elapsed when MCTMPSKP = ‘0’.
- When all the sub-system clocks are ready, then the RCC sets the bit CKREST to ‘0’. The registers listed on *Table 55* can then be modified by the application if needed.
- If the system exits from one of the Stop modes due to a wakeup event for the MCU, when **mcuss_ck,** and **axiss_ck** (if needed) clocks are ready, then the clocks can be re-enabled. The AXICKRDY bit is set to ‘1’ by the RCC if **axiss_ck** has been enabled as well.
- If the system exits from one of the Stop modes due to a wakeup event for the MPU, when **mpuss_ck, mcuss_ck,** and **axiss_ck** clocks are ready, then the clocks can be re-enabled.The AXICKRDY bit is set to ‘1’ as well.

Refer to *Section 10.6.5: Clock restore sequence examples* for additional information.

Figure 69. Clock restore process activated after system Stop

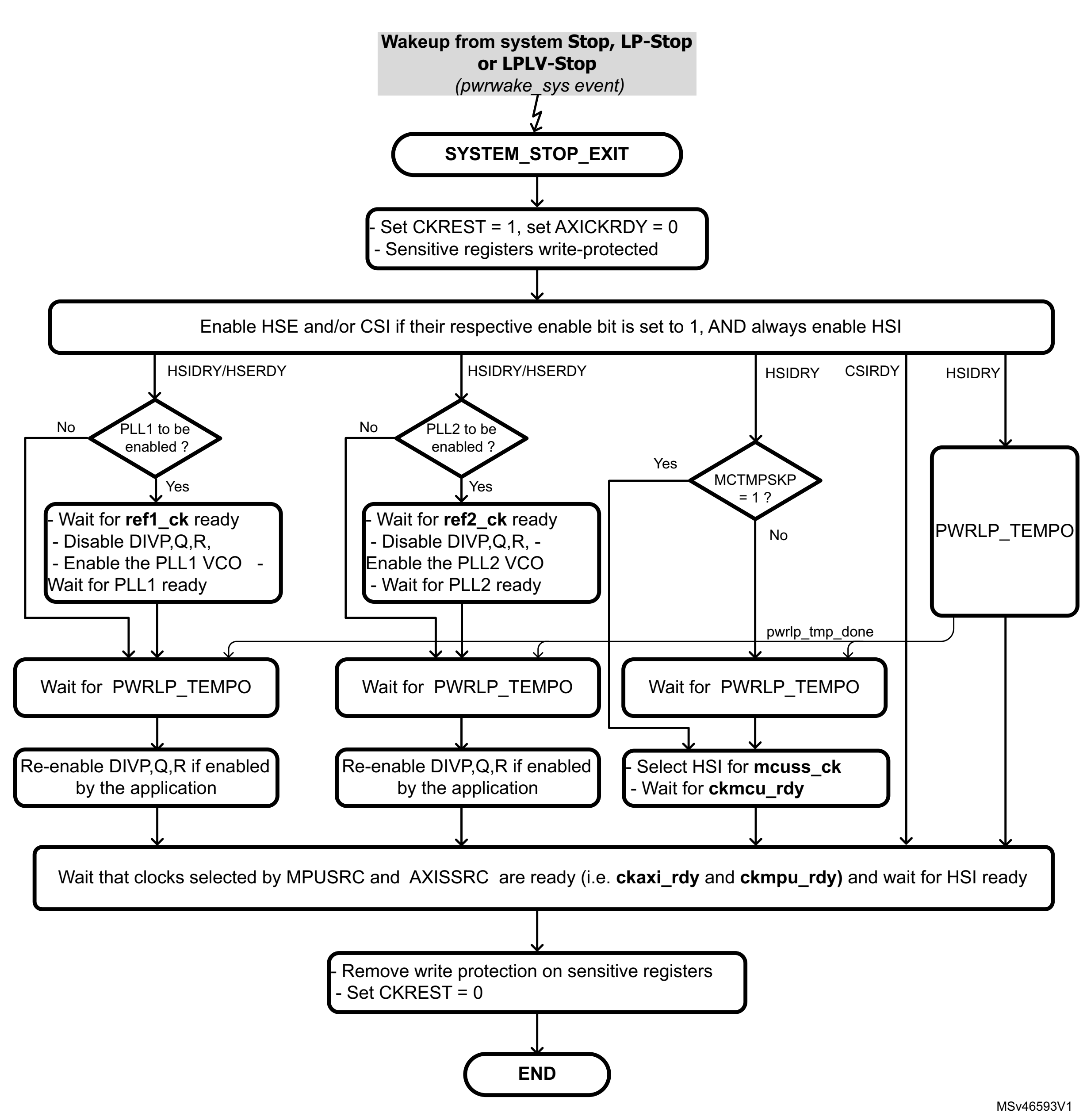
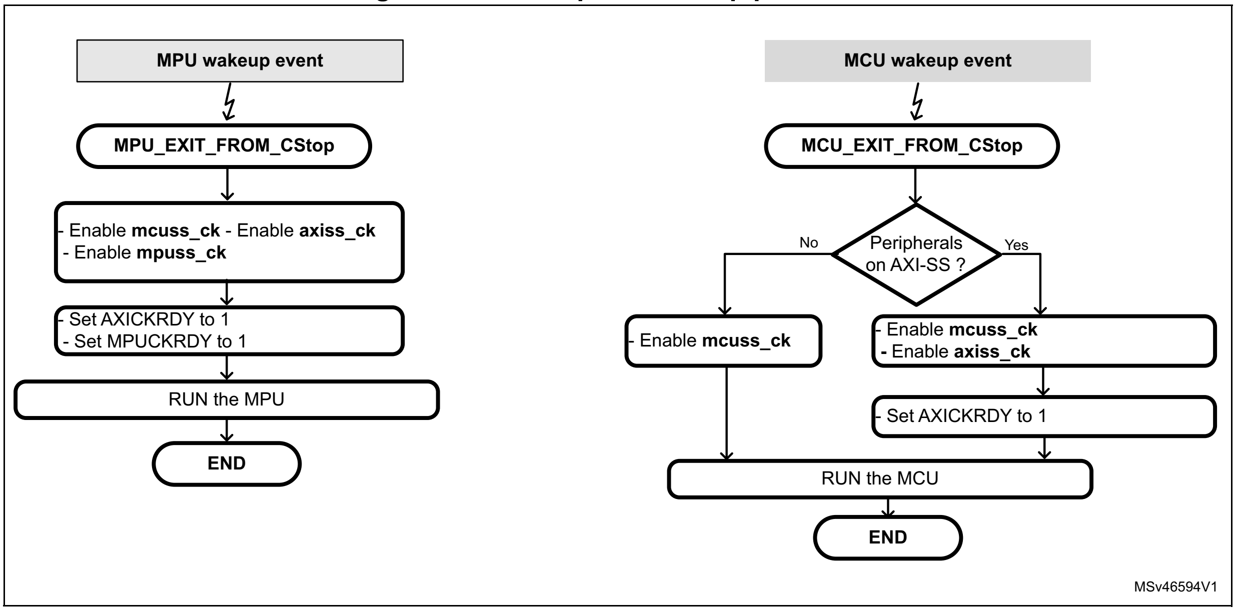


Figure 70. Wake up from CStop processes



Note: Note as well that the RCC also provides a bit named AXICKRDY indicating if the ***axiss_ck*** *clock is ready to be used. This signal is particularly helpful when the MCU allocates a peripheral located into the AXI sub-system.*

Note: When the system exits from one of the Stop modes due to a wakeup event for the MCU, if the MCU has a peripheral using HSE as kernel clock, the MCU has to check that the HSE is ready before using this peripheral.

In order to avoid any unexpected behavior during the clock restore, the RCC provides a write-protection of all registers having a sensitive content for the clock restore. The application shall not attempt to write those registers when the clock restore sequence is running. If a processor attempts to write into one of the register listed in *Table 55*, the write operation is ignored, and a hard fault is generated. This situation may occur if one processor is already running while the other exits from CStop.

Note that the RCC provides a bit named CKREST in order to inform the application if a clock restore is on-going.

Table 55. Resources blocked during clock restore sequence

Resources	-	Corresponding registers write-protected during clock restore
Sub-system clock switches	→	RCC_MPCKSELR, RCC_ASSCKSELR, RCC_MSSCKSELR
PLL1 and 2 settings	→	RCC_PLL1CR, RCC_PLL1CFGR1, RCC_PLL1CFGR2, RCC_PLL1FRACR, RCC_PLL1CSGR RCC_PLL2CR, RCC_PLL2CFGR1, RCC_PLL2CFGR2, RCC_PLL2FRACR, RCC_PLL2CSGR
Reference clocks selection	→	RCC_RCK12SELR
Oscillator settings	→	RCC_OCENSETR, RCC_OCENCLRR, RCC_HSICFGR, RCC_CSICFGR
Miscs	→	RCC_PWRLPDLYCR

Going to Standby mode

The system can go to Standby mode when MPU is in CStop or CStandby, and the MCU is in CStop and both processors allowed the Standby mode. Refer to *Section 9.6: PWR low-power modes* for details

When the complete system goes to Standby mode, the RCC stops all the clocks and PLLS, with the exception of the LSI and LSE.

In addition, the RCC informs the PWR block that the MPU and MCU sub-system are in CStop via **pwrds_mpu** and **pwrds_mcu** signals (see *Figure 46*).

Then the PWR block can request to the external PMIC to switch off the V_{DDCORE} voltage. Only parts working on V_{DD} and V_{SW} are kept operating in this mode.

Leaving Standby mode

When the circuit leaves the system Standby mode, the HSI is selected as default clock for **mpuss_ck**, **axiss_ck**, **mcuss_ck**, the RCC registers are reset to their initial values except the RCC_MP_BOOTCR, RCC_BDCR, RCC_RDLSICR, RCC_BR_RSTSCLRR and RCC_MC_RSTSCLRR.

Note that a system reset is generated when the circuit exits from Standby.

10.4.8 Peripheral clock distribution

Some peripherals are designed to work with two different clock domains asynchronous each other: a clock domain synchronous to the register and bus interface (**ckg_bus_perx** clock), and a clock domain generally synchronous to the peripheral (kernel clock).

The benefit of having peripherals supporting those two clock domains is that the user has more freedom to choose optimized clock frequency for the CPUs, bus matrix, and for the kernel part of the peripheral.

As a consequence, the user can change the bus frequency without reprogramming the peripherals. For example an on-going transfer with UART will not be disturbed if its APB clock is changed on-the-fly.

For most of the peripherals, *Table 56* displays the peripheral clocks under RCC control. Note that clocks controlled by the peripherals themselves are not shown (for examples clocks from an external interfaces and the peripherals).

Table 56 show clearly (gray shaded cells) the kernel clocks provided by the RCC, and the bit field controlling the kernel clock selector. The maximum allowed frequency is given for the output of each MUX and for each bus clock.

It is up to the user to respect those requirements.

Table 56. Peripheral clock distribution overview

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
ADC12	Kernel	←	A	133 (2)	ADCSRC	0	pll4_r_ck
						1	per_ck
						2	pll3_q_ck
	Bus	←	-	209	-	-	hclk2
BSEC	Bus	←	-	67	-	-	pclk5
CRC1	Bus	←	-	266	-	-	hclk6
CRC2	Bus	←	-	209	-	-	hclk3
CRYP1	Bus	←	-	266	-	-	hclk5
CRYP2	Bus	←	-	209	-	-	hclk3
DAC1 and 2	Kernel	←	A	10	-	-	lsi_ck
	Bus	←	-	104.5	-	-	pclk1
DCMI	Bus	←	-	209	-	-	hclk3
DDRC	Kernel	←	A	533	-	-	ker_ddrc_ck
	Bus	←	-	133	-	-	pclk4
				266	-	-	aclk
DDRPERFM	Kernel	←	A	533	-	-	ker_ddrc_ckg
	Bus	←	-	133	-	-	pclk4
DDRPHYC	Kernel	←	A	533	-	-	pll2_r_ck
	Bus	←	-	133	-	-	pclk4
DFSDM	Kernel	←	A	133	SAI1SRC	0	pll4_q_ck
						1	pll3_q_ck
						2	I2S_CKIN
						3	per_ck
						4	pll3_r_ck
		←	S	209	-	-	mlhclk
	Bus	←	-	104.5	-	-	pclk2
DLYBSD1, DLYBSD2	Bus	←	-	266	-	-	hclk6
DLYBSD3	Bus	←	-	209	-	-	hclk2
DLYBQS	Bus	←	-	266	-	-	hclk6
DMA1, DMA2	Bus	←	-	209	-	-	mlhclk
							hclk1
							hclk2
DMAMUX	Bus	←	-	209	-	-	hclk2

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
ETH	Kernel	←	A	125	ETHSRC	0	pll4_p_ck
						1	pll3_q_ck
	Bus	←	-	266	-	-	hclk6
							aclk
ETZPC	Bus	←	-	133	-	-	pclk5
EXTI	Bus	←	-	209	-	-	hclk4
FDCAN	Kernel	←	A	100	FDCANSRC	0	hse_ker_ck
						1	pll3_q_ck
						2	pll4_q_ck
						3	pll4_r_ck
	Bus	←	-	104.5	-	-	pclk2
FMC	Kernel	←	A	266	FMCSRC	0	aclk
						1	pll3_r_ck
						2	pll4_p_ck
						3	per_ck
	Bus	←	-	266	-	-	hclk6
							aclk
GPIOA-K	Bus	←	-	209	-	-	hclk4
GPIOZ	Bus	←	-	266	-	-	hclk5
HASH1	Bus	←	-	266	-	-	hclk5
HASH2	Bus	←	-	209	-	-	hclk3
HDMI-CEC	Kernel	←	-	1	CECSRC	0	lse_ck
						1	lsi_ck
						2	csi_ker_ck/122
	Bus	←	-	104.5	-	-	pclk1
HDP	Bus	←	-	104.5	-	-	pclk3
HSEM	Bus	←	-	209	-	-	hclk3
I2C1, I2C2 I2C3, I2C5	Kernel	←	A	104.5	I2C12SRC, I2C35SRC	0	pclk1
						1	pll4_r_ck
						2	hsi_ker_ck
						3	csi_ker_ck
	Bus	←	-	104.5	-	-	pclk1

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
I2C4, I2C6	Kernel	←	A	133	I2C46SRC	0	pclk5
						1	pll3_q_ck
						2	hsi_ker_ck
						3	csi_ker_ck
	Bus	←	-	133	-	-	pclk5
IPCC	Bus	←	-	209	-	-	hclk3
IWDG1	Kernel	←	A	1	-	-	lsi_ck
	Bus	←	-	133	-	-	pclk5
IWDG2	Kernel	←	A	1	-		lsi_ck
	Bus	←	-	133	-	-	pclk4
LPTIM1	Kernel	←	A	104.5	LPTIM1SRC	0	pclk1
						1	pll4_p_ck
						2	pll3_q_ck
						3	lse_ck
						4	lsi_ck
						5	per_ck
	Bus	←	-	104.5	-	-	pclk1
LPTIM2, LPTIM3	Kernel	←	A	104.5	LPTIM23SRC	0	pclk3
						1	pll4_q_ck
						2	per_ck
						3	lse_ck
						4	lsi_ck
	Bus	←	-	104.5	-	-	pclk3
LPTIM4, LPTIM5	Kernel	←	A	104.5	LPTIM45SRC	0	pclk3
						1	pll4_p_ck
						2	pll3_q_ck
						3	lse_ck
						4	lsi_ck
						5	per_ck
	Bus	←	-	104.5	-	-	pclk3
LTDC	Kernel	←	A	90	-	-	pll4_q_ck
	Bus	←	-	133	-	-	pclk4
				266	-	-	aclk

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
MDIO	Bus	←	-	104.5	-	-	pclk1
MDMA	Bus	←	-	266	-	-	hclk6
							aclk
PWR	Bus	←	-	209	-	-	hclk4
QUADSPI	Kernel	←	A	266	QSPISRC	0	aclk
						1	pll3_r_ck
						2	pll4_p_ck
						3	per_ck
	Bus	←	-	266	-	-	hclk6
							aclk
RNG1	Kernel	←	A	48	RNG1SRC	0	csi_ker_ck
						1	pll4_r_ck
						2	lse_ck
						3	lsi_ck
	Bus	←	-	266	-	-	hclk5
RNG2	Kernel	←	A	48	RNG2SRC	0	csi_ker_ck
						1	pll4_r_ck
						2	lse_ck
						3	lsi_ck
	Bus	←	-	209	-	-	hclk3
RTC/AWU(3)	Kernel	←	A	4	RTCSRC	1	lse_ck
						2	lsi_ck
						3	hse_ker_ck / (RTCDIV+1)
	Bus	←	-	133	-	-	pclk5
SAI1, SAI3	Kernel	←	A	133	SAI1SRC, SAI3SRC	0	pll4_q_ck
						1	pll3_q_ck
						2	I2S_CKIN
						3	per_ck
						4	pll3_r_ck
	Bus	←	-	104.5	-	-	pclk2

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
SAI2	Kernel	←	A	133	SAI2SRC	0	pll4_q_ck
						1	pll3_q_ck
						2	I2S_CKIN
						3	per_ck
						4	spdif_ck_symb
						5	pll3_r_ck
	Bus	←	-	104.5	-	-	pclk2
SAI4	Kernel	←	A	133	SAI4SRC	0	pll4_q_ck
						1	pll3_q_ck
						2	I2S_CKIN
						3	per_ck
						4	pll3_r_ck
	Bus	←	-	104.5	-	-	pclk3
SDMMC1, SDMMC2	Kernel	←	A	200	SDMMC12SRC	0	hclk6
						1	pll3_r_ck
						2	pll4_p_ck
						3	hsi_ker_ck
	Bus	←	-	266	-	-	hclk6
SDMMC3	Kernel	←	A	209	SDMMC3SRC	0	hclk2
						1	pll3_r_ck
						2	pll4_p_ck
						3	hsi_ker_ck
	Bus	←	-	209	-	-	hclk2
SPDIFRX	Kernel	←	A	200	SPDIFSRC	0	pll4_p_ck
						1	pll3_q_ck
						2	hsi_ker_ck
	Bus	←	-	104.5	-	-	pclk1

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
SPI6	Kernel	←	A	133	SPI6SRC	0	pclk5
						1	pll4_q_ck
						2	hsi_ker_ck
						3	csi_ker_ck
						4	hse_ker_ck
						5	pll3_q_ck
	Bus	←	-	133	-	-	pclk5
SPI2S1	Kernel	←	A	200	SPI1SRC	0	pll4_p_ck
						1	pll3_q_ck
						2	I2S_CKIN
						3	per_ck
						4	pll3_r_ck
	Bus	←	-	104.5	-	-	pclk2
SPI4, SPI5	Kernel	←	A	133	SPI45SRC	0	pclk2
						1	pll4_q_ck
						2	hsi_ker_ck
						3	csi_ker_ck
						4	hse_ker_ck
	Bus	←	-	104.5	-	-	pclk2
SPI2S3, SPI2S2	Kernel	←	A	200	SPI23SRC	0	pll4_p_ck
						1	pll3_q_ck
						2	I2S_CKIN
						3	per_ck
						4	pll3_r_ck
	Bus	←	-	104.5	-	-	pclk1
STGEN	Kernel	←	A	64	STGENSRC	0	hsi_ker_ck
						1	hse_ker_ck
	Bus	←	-	133	-	-	pclk4
							pclk5
SYSCFG	Bus	←	-	104.5	-	-	pclk3
DTS	Kernel	←	A	10	-	-	lse_ck
	Bus	←	-	104.5	-	-	pclk3

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
TIM1,TIM8, TIM[17:15]	Kernel	←	S	209	-	-	timg2_ck
	Bus	←	-	104.5	-	-	pclk2
TIM[7:2], TIM[14:12]	Kernel	←	S	209	-	-	timg1_ck
	Bus	←	-	104.5	-	-	pclk1
TZC	Bus	←	-	133	-	-	pclk5
				266	-	-	aclk
USART1	Kernel	←	A	133	UART1SRC	0	pclk5
						1	pll3_q_ck
						2	hsi_ker_ck
						3	csi_ker_ck
						4	pll4_q_ck
						5	hse_ker_ck
	Bus	←	-	133	-	-	pclk5
USART2, UART4 USART3, UART5 UART7, UART8	Kernel	←	A	104.5	UART24SRC, UART35SRC, UART78SRC	0	pclk1
						1	pll4_q_ck
						2	hsi_ker_ck
						3	csi_ker_ck
						4	hse_ker_ck
	Bus	←	-	104.5	-	-	pclk1
USART6	Kernel	←	A	104.5	UART6SRC	0	pclk2
						1	pll4_q_ck
						2	hsi_ker_ck
						3	csi_ker_ck
						4	hse_ker_ck
	Bus	←	-	104.5	-	-	pclk2
USBH	Kernel	←	A	60	-	-	rcc_ck_usbh_free
				60			rcc_ck_usbh_60m
				48			rcc_ck_usbh_48m
				12			rcc_ck_usbh_12m
	Bus	←	-	266	-	-	hclk6

Table 56. Peripheral clock distribution overview (continued)

Peripherals	Clock types	-	Type (1)	Max. freq. [MHz]	Clock MUX	MUX Pos.	Clock sources
USBPHY	Kernel	←	A	38.4	USBPHYSRC	0	hse_ker_ck
						1	pll4_r_ck
						2	hse_ker_ck / 2
	Bus	←	-	133	-	-	pclk4
USBO	Kernel	←	A	50	USBOSRC	0	pll4_r_ck
						1	rcc_ck_usbo_48m
	Kernel	←	A	60	-	-	rcc_ck_usbo_60m
	Bus	←	-	209	-	-	hclk2
VREFBUF	Bus	←	-	104.5	-	-	pclk3
WWDG1	Bus	←	-	104.5	-	-	pclk1

1. '**A**' Means that the kernel clock is asynchronous with respect to bus interface clock.
   '**S**' Means that the kernel clock is synchronous with respect to bus interface clock.
2. For the ADC12, if a clock with a duty cycle of 50% is needed then the max frequency is limited to 72 MHz. In order to get a duty cycle 50%, the DIVQ value shall be odd. Refer to *Section : Clock distribution for HDMI-CEC, STGEN, ADCs and DACs* for details.
   For the SDMMCx, a duty cycle of 50% in needed when supporting DDR, if the RCC cannot provide a clock with such a duty cycle, the application must use SDMMCx internal dividers.
3. The RTC switch is in the V_{SW} voltage domain

The following sections show a more detailed view of the kernel clock and bus interface clock distribution. Refer to *Section 10.4.11: Peripheral clock gating control* for details.

In order to reduce the amount of switches, some peripherals are sharing the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

Clock distribution for SAIs, SPI/I2Ss, SPDIFRX and DFSDM

The audio peripherals generally need specific accurate frequencies, except for SPDIFRX. As shown in *Figure 72* and *Figure 73*, the kernel clock of the SAIs or SPI(I2S)s can be generated by:

- The PLL3 or PLL4 in order to provide optimal flexibility on the frequency generation,
- HSE, HSI or CSI for use-cases where the current consumption is critical,
- I2S_CKIN in the case where an external clock reference need to be used.The DFSDM can share the same kernel clock as SAI1: this can be useful when the DFSDM is used for audio applications.

To work properly, the DFSDM always needs **dfsdm_clk** and **fdsdm_pclk** clocks.

The clock **dfsdm_Aclk** can be used as clock source for DFSDM_CKOUT. The clock DFSDM_CKOUT can be generated either from **dfsdm_clk** (default) or **dfsdm_Aclk**.

See *Section 32: Digital filter for sigma delta modulators (DFSDM)* for details.

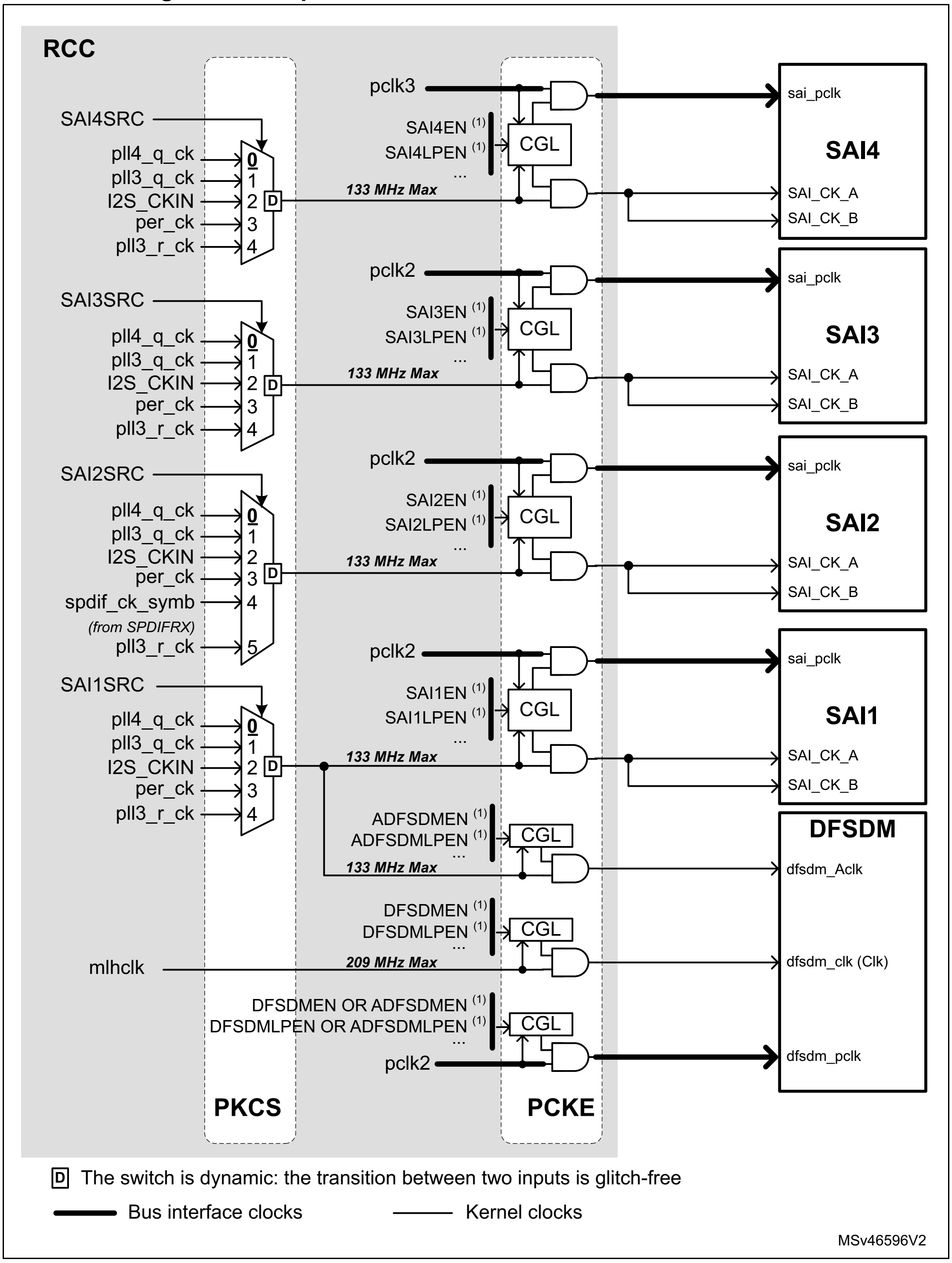
Clock settings examples for PLL3

The table hereafter is showing programming examples of kernel clock generation using the PLL3. Audio frequencies with good accuracy (grey cells) can be obtained with the PLL3 in fractional or integer mode. The PLL3 is also used to provide a clock to the MCU (PLL3P output).

Table 57. Kernel clock settings examples for audio applications

Conf	F_{HSE}	F_{ref3_ck}	F_{VCO}	DIVM+1	DIVN+1	FRACV	DIVP+1	DIVQ+1	DIVR+1	PLL3P output	PLL3Q output		PLL3R output	
											Freq.	Error for 48 kHz	Freq.	Error for 44.1 kHz
#1	24	12	417.71	2	34	6632	2	17	37	208.9	24.5715	-185	11.2896	-1
#2	24	12	417.79	2	34	6685	2	17	37	208.9	24.5760	1	11.2917	185
#3	24	12	417.75	2	34	6659	2	17	37	208.9	24.5738	-90	11.2907	94
#4	24	4.8	614.40	5	128	0	4	25	-	153.6	24.576	0	-	-
#5	24	4.8	609.6	5	127	0	4	-	27	152.4	-	-	22.5778	-63
-	MHz	MHz	MHz	-	-	-	-	-	-	MHz	MHz	ppm	MHz	ppm

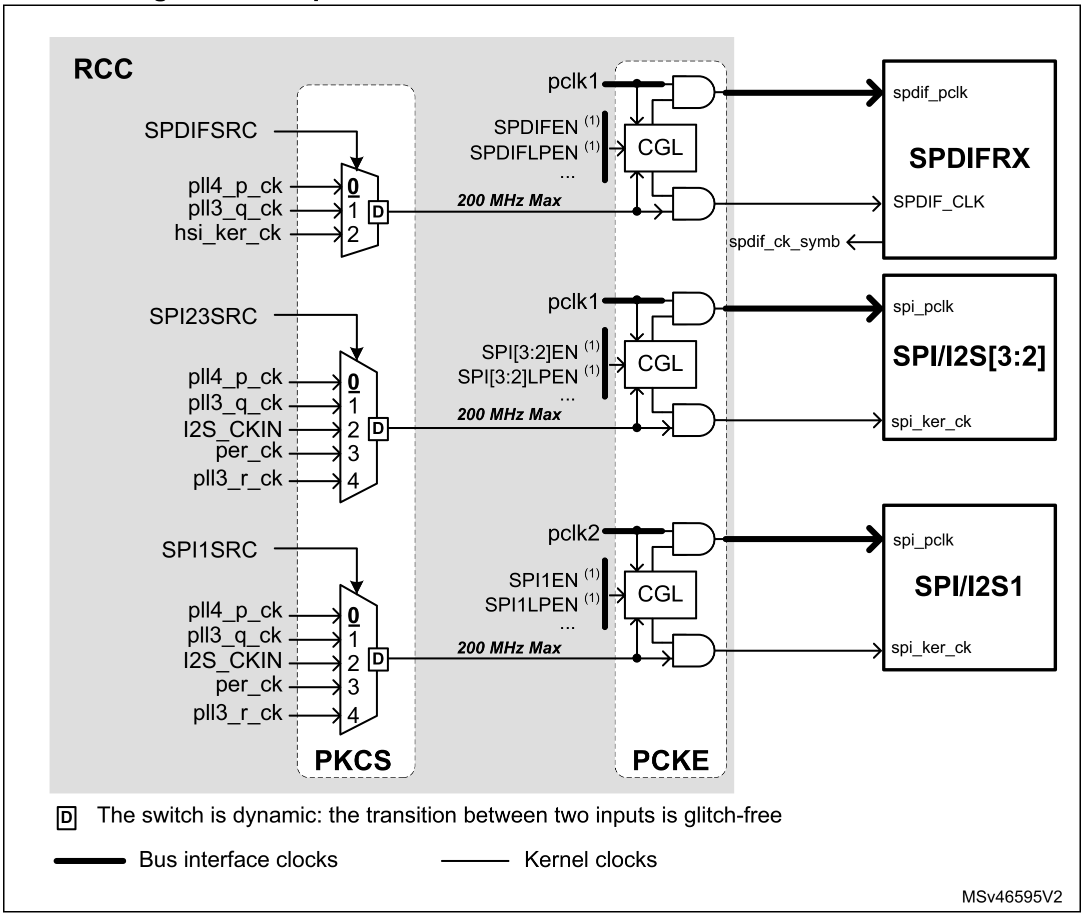
Figure 71. Peripheral clock distribution For SAIs and DFSDM



1. There is one enable bit and one low-power enable bit for MPU and MCU.
   X Represents the selected MUX input after a system reset

Note: The SPDIFRX does not need a specific frequency, but just a kernel clock frequency high enough to make the peripheral working properly. Refer to the description of the SPDIFRX for details.

Figure 72. Peripheral clock distribution for SPI/I2Ss and SPDIFRX



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

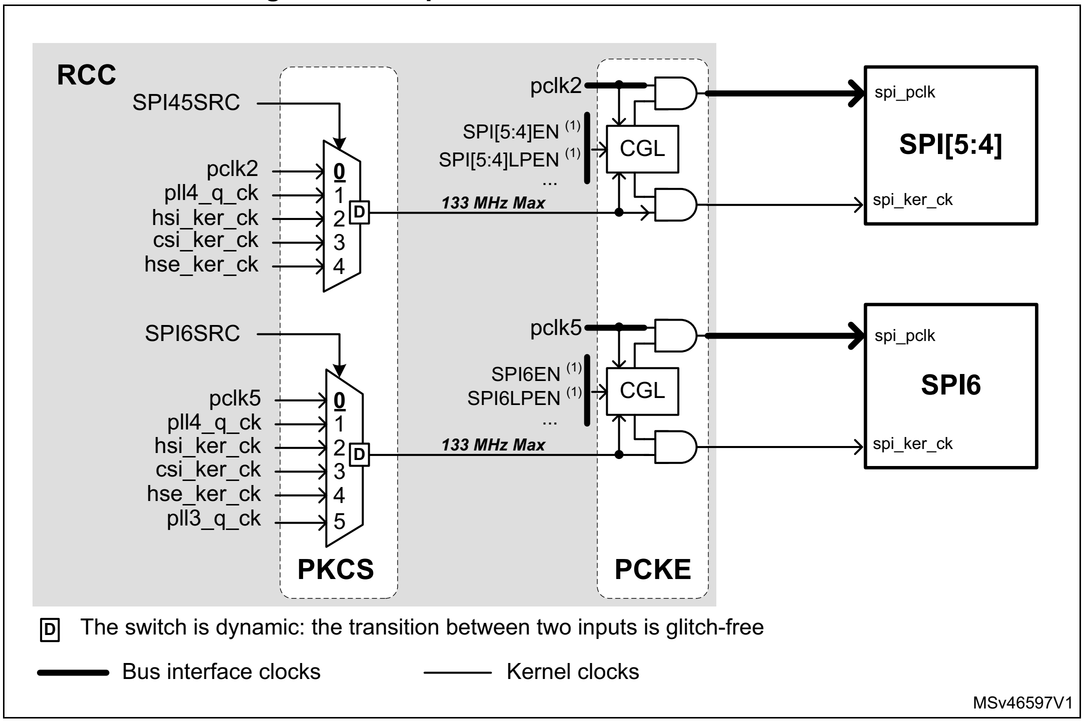
Refer to *Table 63* for details concerning the behavior of the clock gating logic (CGL).

Clock distribution for SPIs

In general, SPIs (or SPI/I2Ss used in SPI mode) do not need specific kernel clock frequencies but a clock fast enough to generate the correct data rate. For that purpose it is generally possible to select a source from:

- APB bus, which can be used to avoid the activation of an additional PLL,
- PLL4 for better flexibility if required, it allows for example to change the frequency of the MCU bus via PLL3, without affecting the speed of some serial interfaces,
- HSI, CSI for low-power use-cases.

Figure 73. Peripheral clock distribution for SPIs



1. There is one enable bit and one low-power enable bit for MPU and MCU.
   X Represents the selected MUX input after a system reset.

*Note: In order to provide a secure kernel clock to SPI6, the application must choose **pclk5**, **hse_ker_ck**, **hsi_ker_ck** or **csi_ker_ck**. The application can also use **pll3_q_ck**, but in that case the bit MCKPROT must be set to '1' as well. Refer to Section 10.4.14: RCC TrustZone function for usage of the MCKPROT bit.*

Refer to *Table 63* for details concerning the behavior of the clock gating logic (CGL).

Clock distribution for I2Cs, UARTs and USARTs

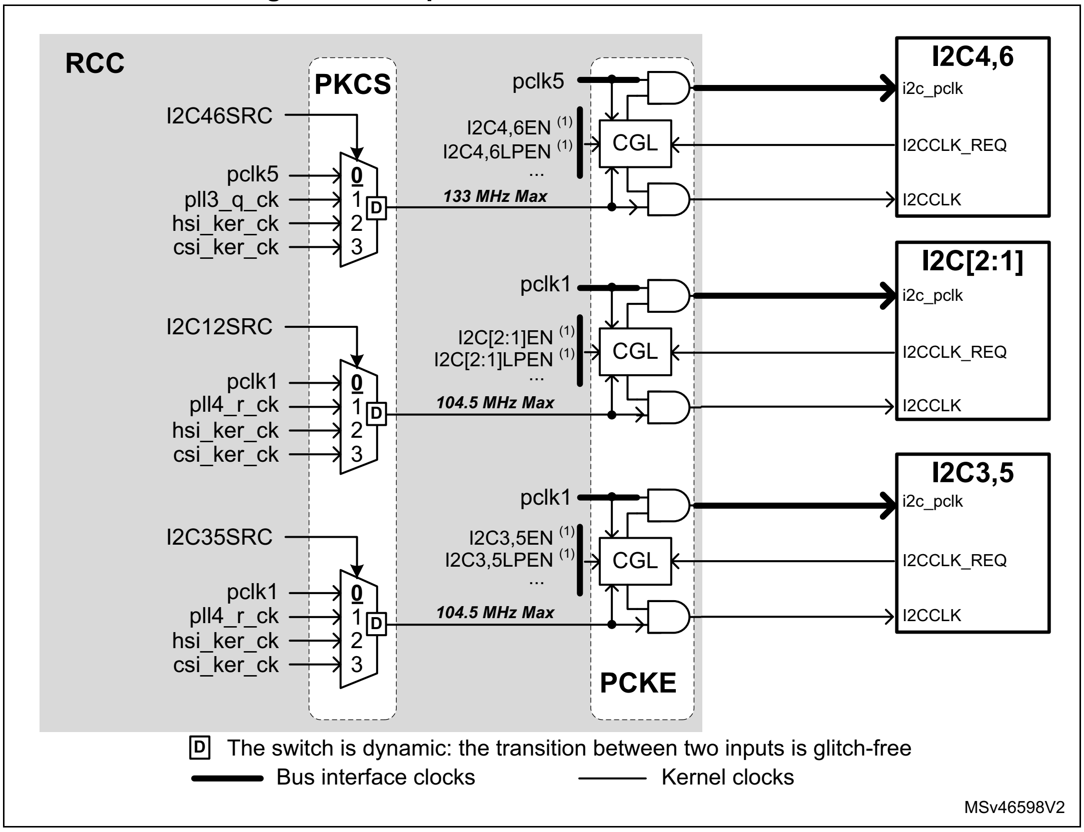
As explained for the SPI, I2Cs and UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the wanted bit clock on the serial link. For that purpose it is generally possible to select a source from:

- PLL3 which can be used to avoid the activation of an additional PLL,
- PLL4 for better flexibility if required, it allows for example to change the frequency of the MCU bus via PLL3, without affecting the speed of some serial interfaces,
- HSI, CSI for low-power use-cases, and also in the case where the peripheral has to wake up quickly from (LP-)Stop mode (i.e. UART, I2C...).

Note that UARTs also receive the LSE clock, for use-cases which do not request high baud rates.

As shown in *Figure 74*, *Figure 75* and *Figure 76* the I2Cs and UARTs provide kernel clock requests signals (I2CCLK_REQ and UCLK_REQ). These signals can be used when the system is in (LP-)Stop, in order to request the selected kernel clock to the RCC. This function only works when the **hsi_ker_ck** or **csi_ker_ck** or **hse_ker_ck** are selected. Refer to *Section 10.4.11: Peripheral clock gating control* for details.

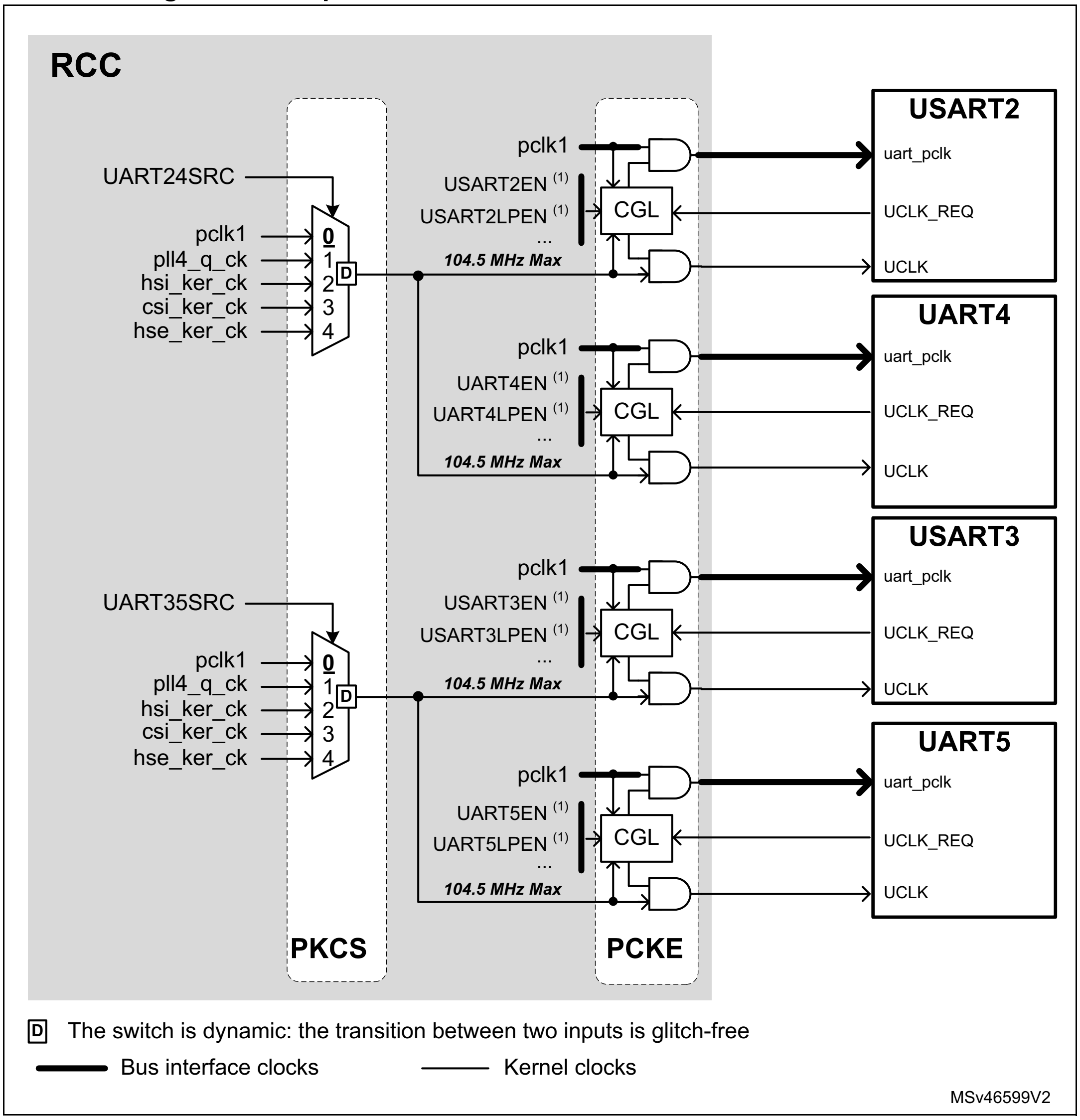
Figure 74. Peripheral clock distribution for I2Cs



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

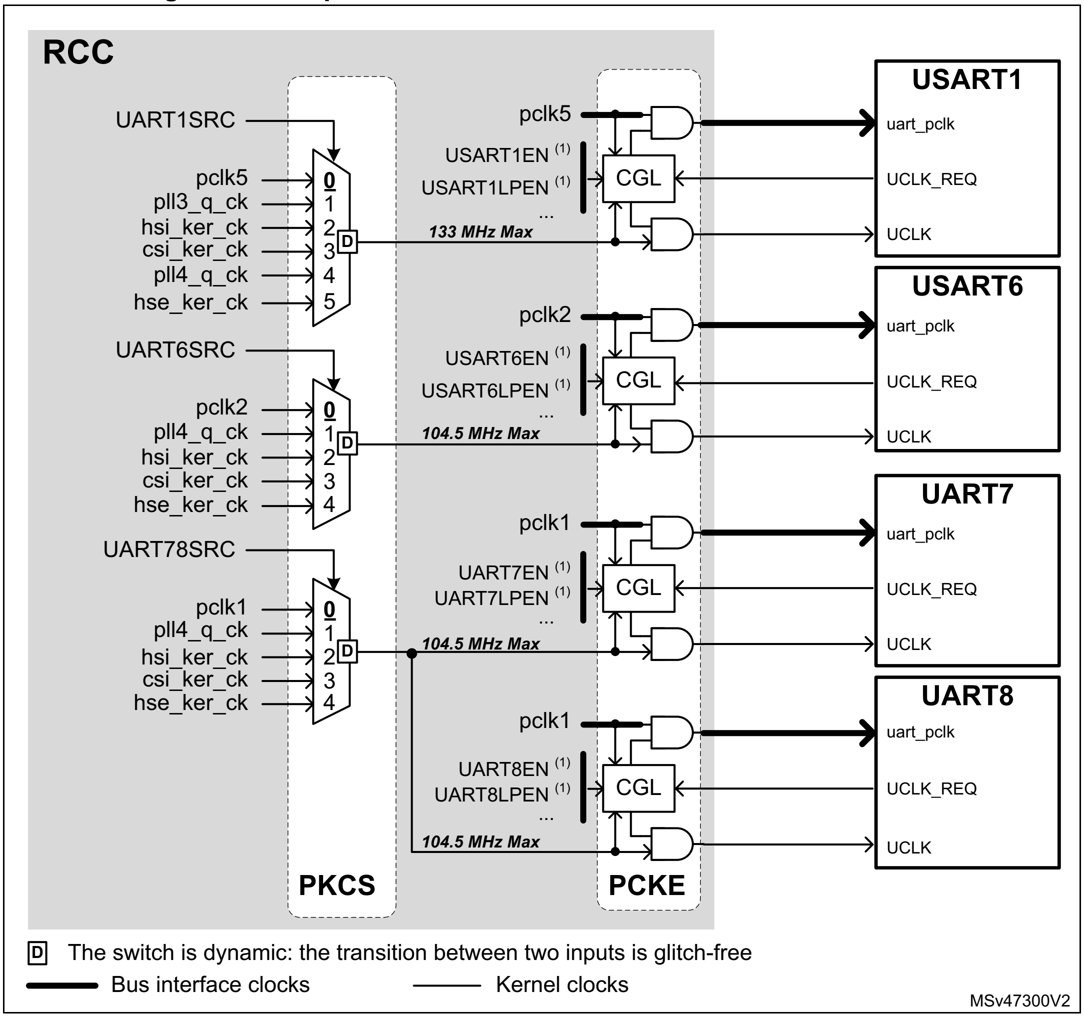
Note: *In order to provide a secure kernel clock to I2C4 and I2C6, the application must choose **pclk5**, **hsi_ker_ck** or **csi_ker_ck**. The application can also use **pll3_q_ck**, but in that case the MCKPROT bit must be set to '1' as well.*

Figure 75. Peripheral clock distribution for UARTs, and USARTs



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

Figure 76. Peripheral clock distribution for UARTs, and USARTs



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

*Note: In order to provide a secure kernel clock to USART1, the application must choose **pclk5**, **hse_ker_ck**, **hsi_ker_ck** or **csi_ker_ck**. The application can also use **pll3_q_ck**, but in that case the MCKPROT bit must be set to '1' as well.*

*Note: In order to allow the I2Cs or U(S)ARTs to work in CStop or system (LP-)Stop mode, the user must select an oscillator as kernel clock: **hse_ker_ck**, **hsi_ker_ck** or **csi_ker_ck**. The HSEKERON, HSIKERON and CSIKERON bits must be programmed according to application requirements.*

Refer to *Table 63* for details concerning the behavior of the clock gating logic (CGL).

Clock distribution for TIMs and LPTIMs

The TIMs timers are using kernel clocks (**timg[2:1]_ck**) phase aligned with the bus interface clock.

The frequency of the timers clock depends on:

- The APB prescaler corresponding to the bus where the timer is connected, and
- The bits TIMG[2:1]PRE.

The next table shows how to select the timer clock frequency.

Table 58. Ratio between clock timers and pclk

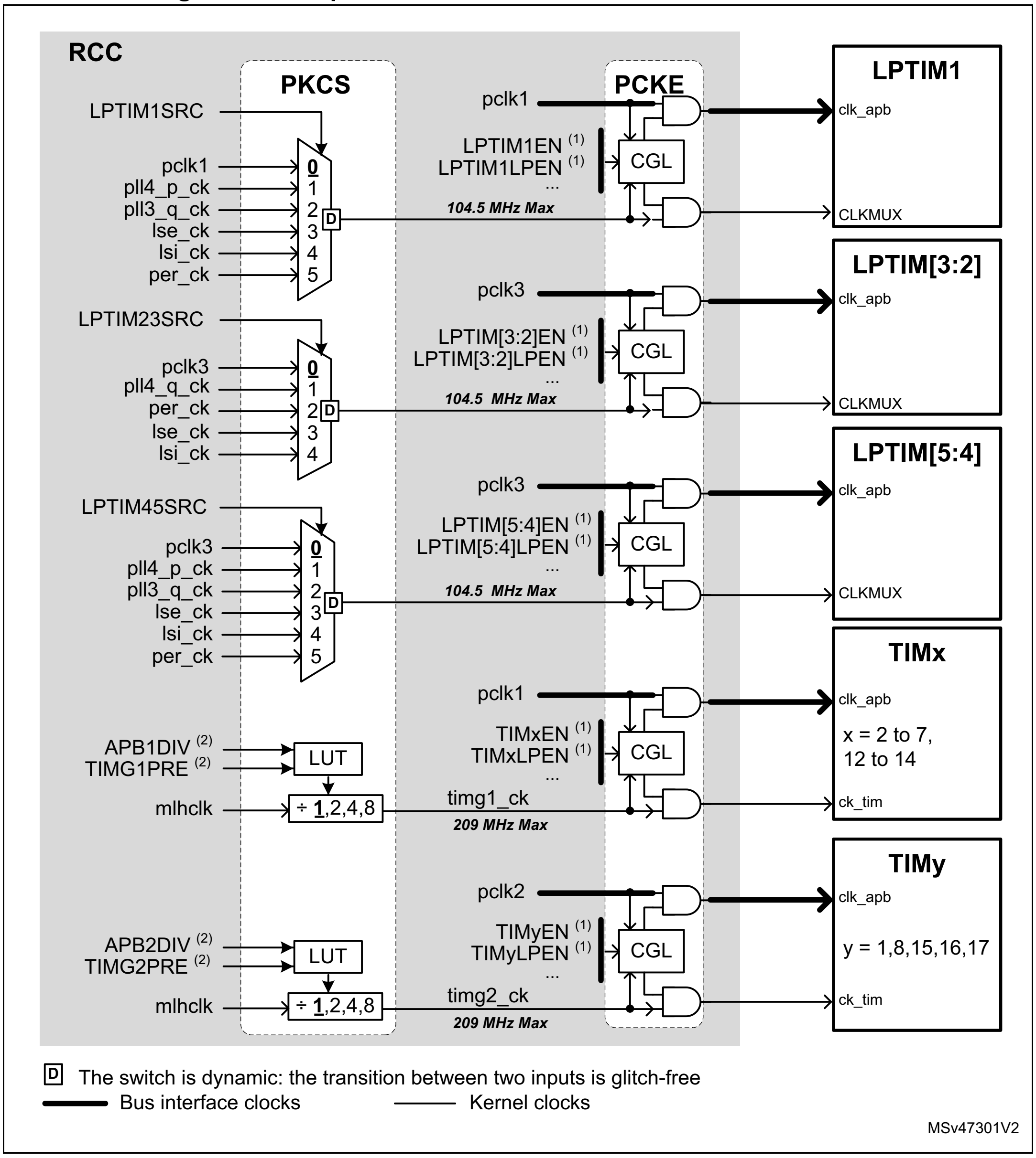
APB1DIV (1) APB2DIV	TIMG1PRE TIMG2PRE (2)	-	F_{timg1_ck} F_{timg2_ck}	F_{pclk1} F_{pclk2}	Comments
‘000’	x	→	F_{mlhclk}	F_{mlhclk}	The timer clock is equal to the bus clock.
‘001’	x	→	F_{mlhclk}	F_{mlhclk} / 2	The timer clock is 2 times faster than the bus clock.
‘010’	‘0’	→	F_{mlhclk} / 2	F_{mlhclk} / 4	
‘011’	‘0’	→	F_{mlhclk} / 4	F_{mlhclk} / 8	
‘100’	‘0’	→	F_{mlhclk} / 8	F_{mlhclk} / 16	
‘010’	‘1’	→	F_{mlhclk}	F_{mlhclk} / 4	The timer clock is 4 times faster than the bus clock.
‘011’	‘1’	→	F_{mlhclk} / 2	F_{mlhclk} / 8	
‘100’	‘1’	→	F_{mlhclk} / 4	F_{mlhclk} / 16	

1. APB1DIV and APB2DIV are located into *RCC APB1 clock divider register (RCC_APB1DIVR)* and *RCC APB2 clock divider register (RCC_APB2DIVR)*.
2. TIMG1PRE and TIMG2PRE are located into *RCC TIM group 1 prescaler register (RCC_TIMG1PRER)* and *RCC TIM group 2 prescaler register (RCC_TIMG2PRER)*.

The LPTIM timers have a kernel clock fully asynchronous with respect to their bus interface clock.

In addition, the LPTIMs can also use as reference clock **lse_ck** and **lsi_ck**, allowing them to work even when the system is in (LP-/LPLV-)Stop.

Figure 77. Peripheral clock distribution for TIMs and LPTIMs



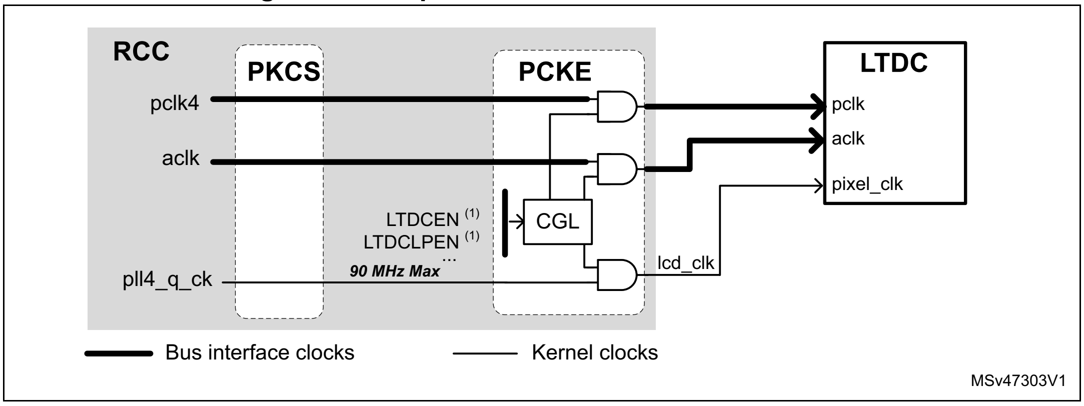
1. There is one enable bit and one low-power enable bit for MPU and MCU.
   X Represents the selected MUX input after a system reset.
2. Can be changed on the fly.

Refer to *Table 63* for details concerning the behavior of the clock gating logic (CGL).

Clock distribution for LTDC

The LTDC pixel clock is provided by the PLL4 (**pll4_q_ck**). The frequency of this clock shall be adjusted according to the wanted resolution and refresh rate.

Figure 78. Peripheral clock distribution for LTDC

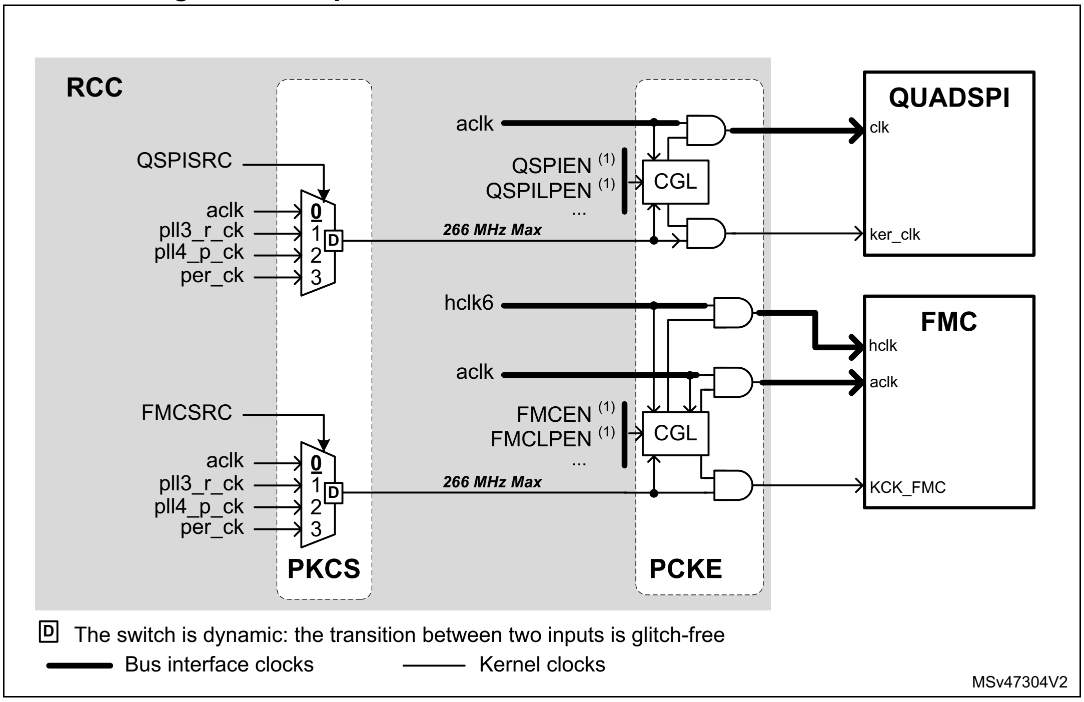


1. There is one enable bit and one low-power enable bit for MPU and MCU.

Clock distribution for SDMMCs, FMC and QUADSPI

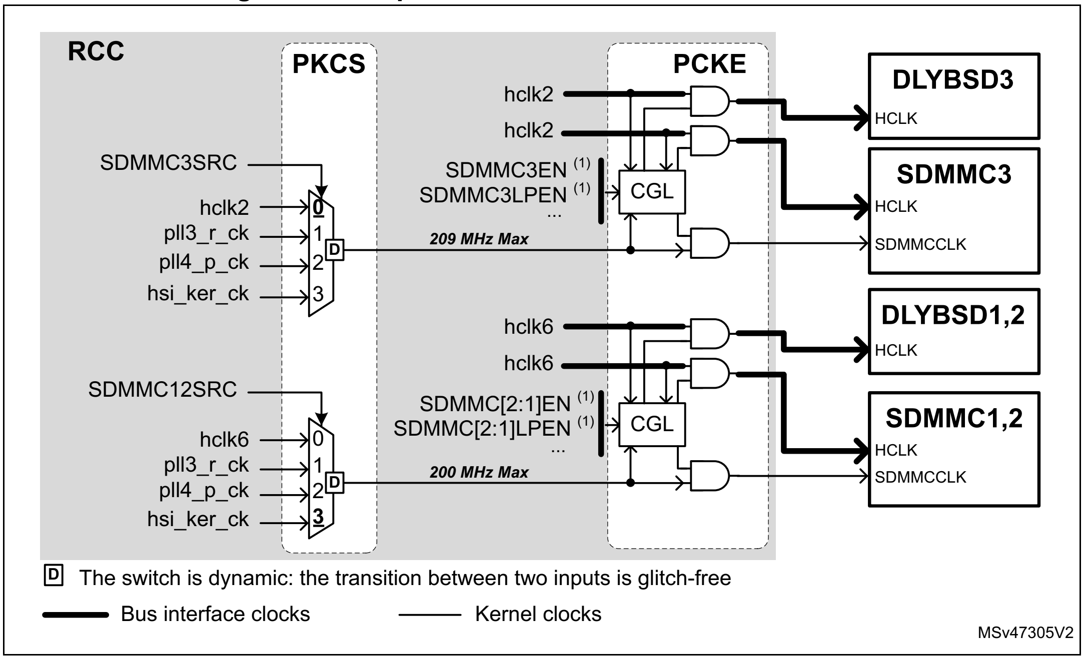
The FMC, QUADSPI can also use a clock different from the bus interface clock for more flexibility.

Figure 79. Peripheral clock distribution for QUADSPI and FMC



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

Figure 80. Peripheral clock distribution for SDMMCs



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

Clock distribution for USBH and USBO

Figure 81 shows the clock distribution for the USB blocks. The RCC has to provide a clean reference clock to the USB PHY. The USB PHY has its own PLL and generates kernel clocks for the USBH (HOST) and USBO (OTG) blocks. The USB PHY receives two different clocks thanks to the mux controlled by USBPHYSRC.

In order to get the best clock quality for the USB, it is better to select the **hse_ker_ck** clock or **hse_ker_ck**/2.

Note that the application has to provide to the USBPHY PLL the following clock reference:

- A clock frequency between 19.2 and 38.4 MHz if the USBPHY PLL is working in integer mode
- A clock frequency between 26 and 38.4 MHz if the USBPHY PLL is working in fractional mode

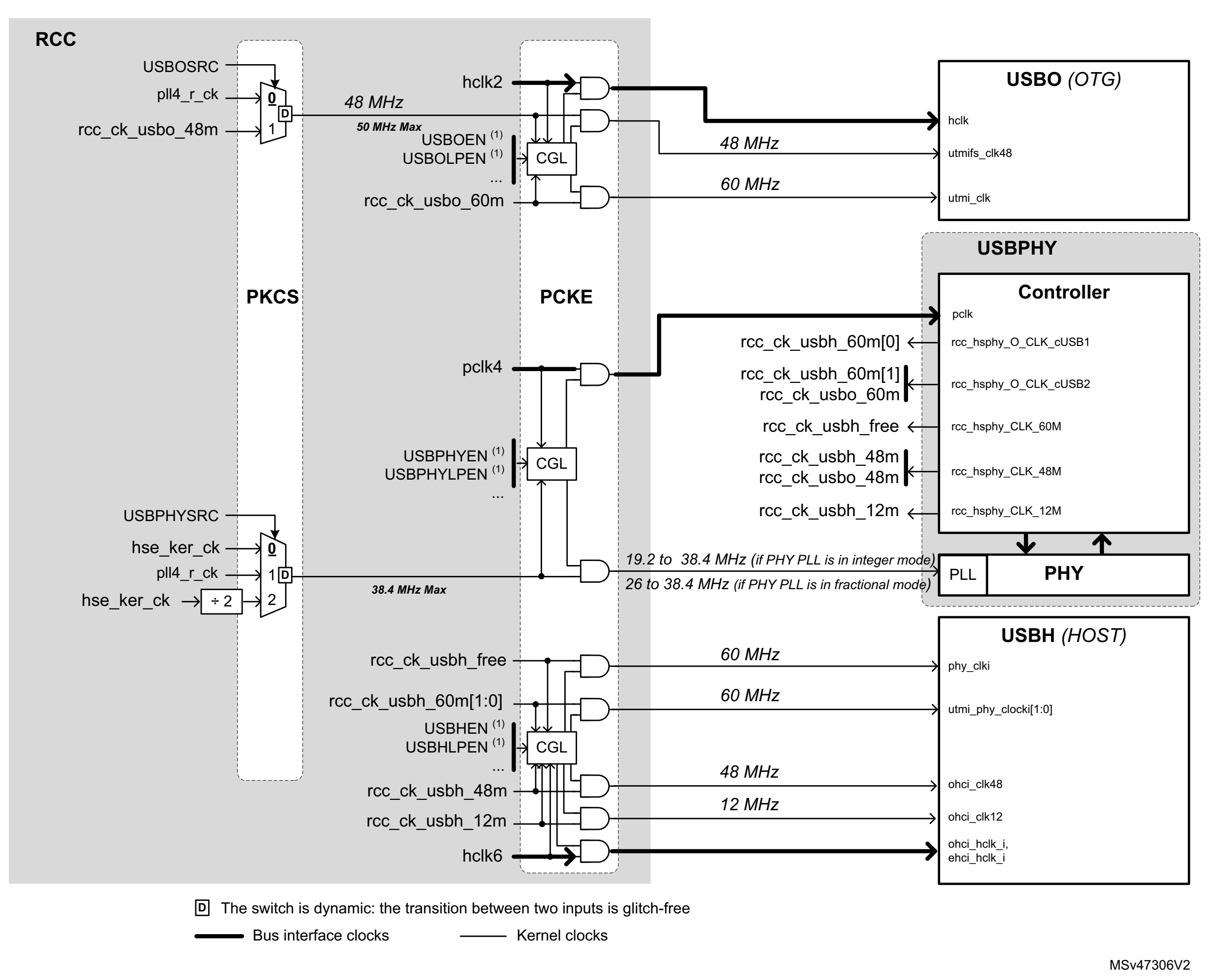
In addition, if the USBO is used in full-speed mode only, the application can choose the 48 MHz clock source to be provided to the USBO:

- A clock generated by the USBPHY; in that case the PLL of the PHY must be used.
- The **pll4_r_ck** clock; in that case there is no need to enable the PLL of the PHY.

The clock selection is performed via USBOSRC.

Those configuration bits are located into *RCC USB kernel clock selection register (RCC_USBCKSELR)*.

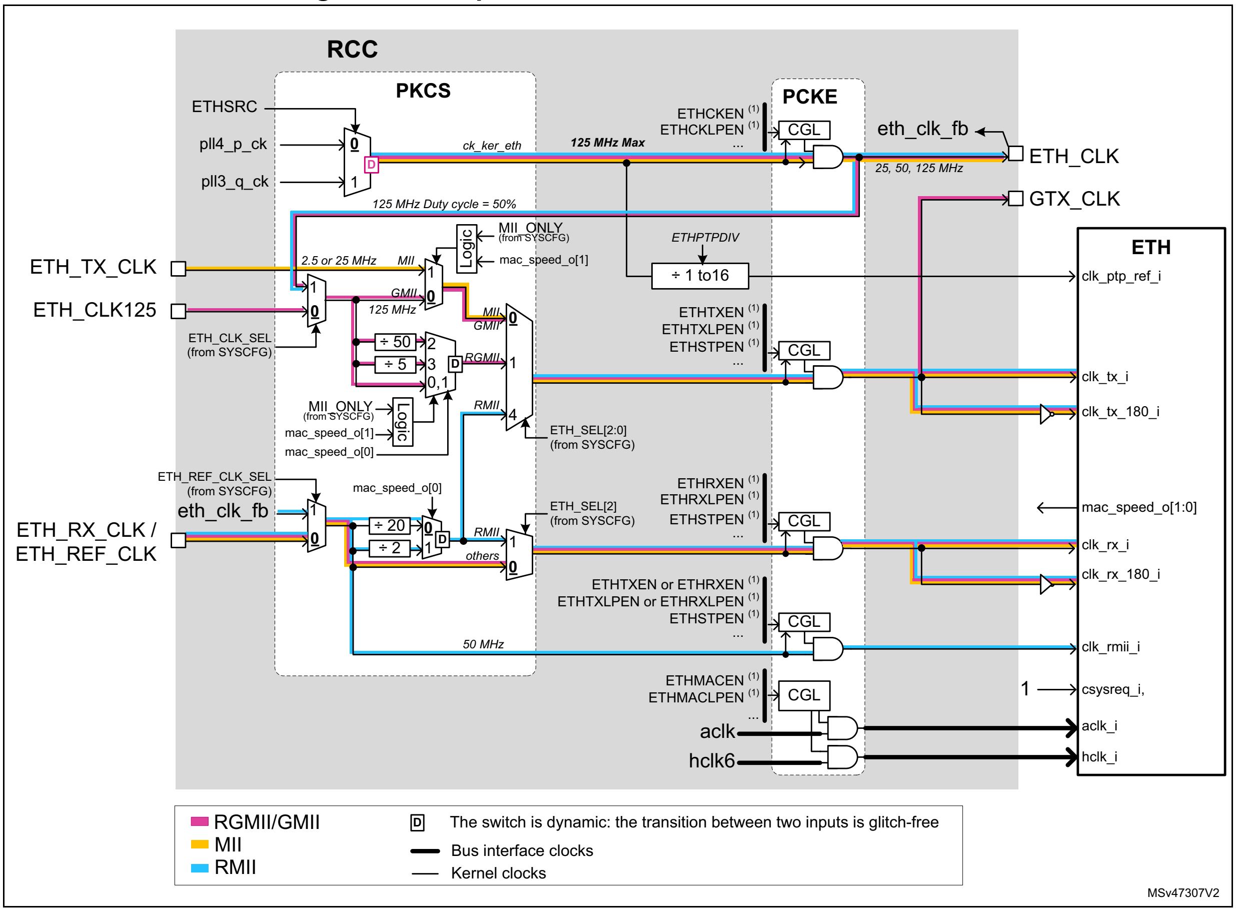
Figure 81. Peripheral clock distribution for USB



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

Clock distribution for Ethernet (ETH)

Figure 82. Peripheral clock distribution for Ethernet



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

The Ethernet clocks provided by the RCC are available on ETH_CLK pad.

The clock selection for the RX and TX path is controlled via the SYSCFG block.

The signal ETH_SEL[2:0] is provided by the SYSCFG block and defines the interface type used:

- In MII mode (orange path),
  - The transmit clock is received from the external PHY via the pad ETH_TX_CLK. The clock frequency will be 2.5 or 25 MHz.
  - The receive clock is received from the external PHY via the pad ETH_RX_CLK. The clock frequency will be 2.5 or 25 MHz
  - In this mode, the output ETH_CLK can be used to provide a reference clock (25 MHz) to the external PHY.
- In GMII mode (red path),
  - The transmit clock can be selected between an internal clock provided by the RCC, or the clock received on the ETH_CLK125 pad. This selection is controlled

via ETH_CLK_SEL from the SYSCFG.
The clock frequency is fixed to 125 MHz.
    - The receive clock is coming directly from the ETH_RX_CLK pad.
The clock frequency is fixed to 125 MHz.
    - The transmit clock is provided to the PHY via the pad GTX_CLK.
    - A reference clock of 25 MHz can also be provided to the external PHY, via ETH_CLK pad.
- In RGMII mode (red path),
    - The transmit clock can be selected between an internal clock provided by the RCC, or the clock received on the pad ETH_CLK125. This selection is controlled via ETH_CLK_SEL from the SYSCFG. The clock frequency can be adjusted dynamically to 2.5, 25 or 125 MHz according to the signal **mac_speed_o[1:0]** provided by the ETH block
    - The receive clock is coming directly from ETH_RX_CLK pad. The clock frequency will be 2.5, 25 or 125 MHz.
    - The transmit clock is provided to the PHY via the pad GTX_CLK.
    - A reference clock of 25 MHz can also be provided to the external PHY, via ETH_CLK pad.
- In RMII mode (blue path)
    - A continuous reference clock (50 MHz) can be selected between an internal clock provided by the RCC, or the clock received on ETH_REF_CLK pad. This selection is controlled via ETH_REF_CLK_SEL from the SYSCFG.
    - The transmit and receive clock frequency can be adjusted dynamically to 2.5 or 25 MHz according to the signal **mac_speed_o[0]** provided by the ETH block.
    - A continuous reference clock (50 MHz) can also be provided to the external PHY, via ETH_CLK pad.

The ETH block provides information on the MAC speed (**mac_speed_o[1:0])**, which may change dynamically.

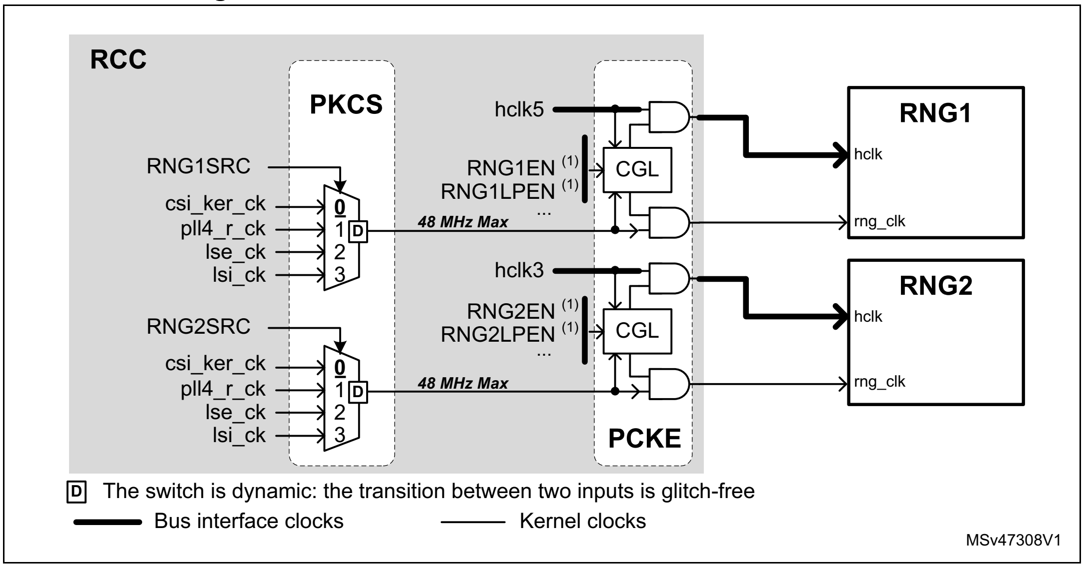
mac_speed_o[1:0] equals:
- 0x: for 1000 Mbps
- 10: for 10 Mbps
- 11: for 100 Mbps

The application can control the gating of the clocks according to the processor modes (CRun, CSleep).

In addition it is also possible to control the gating of clocks coming from PADs ETH_RX_CLK and ETH_TX_CLK in CStop and Stop modes via the ETHSTPEN bit. It is also possible to gate the GTX_CLK clock provided to the PHY during EEE LPI state via the ETH block.

Clock distribution for RNG1 and RNG2

Figure 83. Kernel clock distribution for RNG1 and RNG2

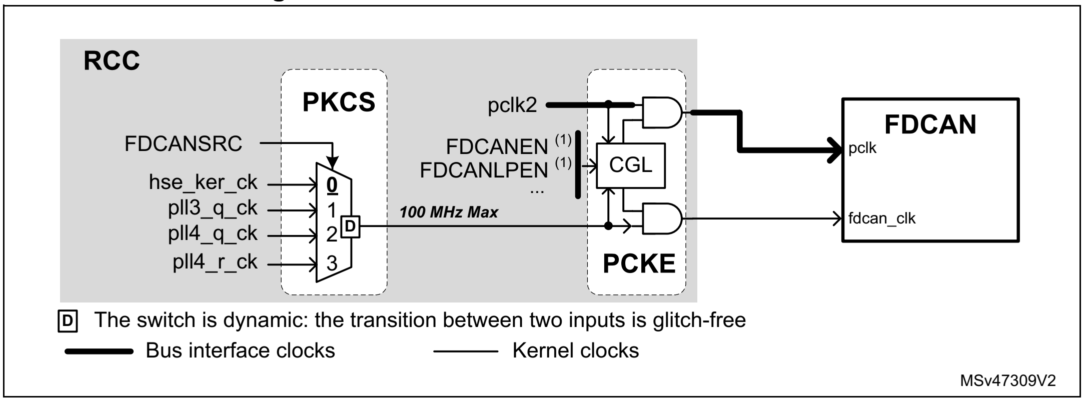


1. There is one enable bit and one low-power enable bit for MPU and MCU.
   X Represents the selected MUX input after a system reset.

Note: In order to provide a secure kernel clock to RNG1, the application must choose ***lse_ck****,* ***lsi_ck****, or* ***csi_ker_ck****.*

Clock distribution for FDCAN

Figure 84. Kernel clock distribution for FDCAN



1. There is one enable bit and one low-power enable bit for MPU and MCU.
   X Represents the selected MUX input after a system reset.

Clock distribution for HDMI-CEC, STGEN, ADCs and DACs

The handling of the STGEN block is a bit particular: It requires a single clock used as bus interface, and kernel clock. In addition, this clock may need to be activated during system (LP-)Stop mode.

Note that two asynchronous bridges are connecting the STGEN to APB4 and APB5 busses, so the RCC shall provide those APB clocks when the STGEN is enabled.

The STGEN can be accessed via two different APB ports:

- The ABP4 port allows the application to use some STGEN resources, but this interface is read-only. Enable bits are available in order to use the STGEN via the APB4: STGENROEN, STGENROLPEN and STGENROSTPEN. These bits can be modified in non-secure mode even if TrustZone is enabled (TZEN = ‘1’).
- The ABP5 port allows the application to control the STGEN, without restrictions. Enable bits are available in order to use the STGEN via the APB5: STGENEN, STGENLPEN and STGENSTPEN. These bits can only be modified by secure accesses if TrustZone is enabled (TZEN = ‘1’).

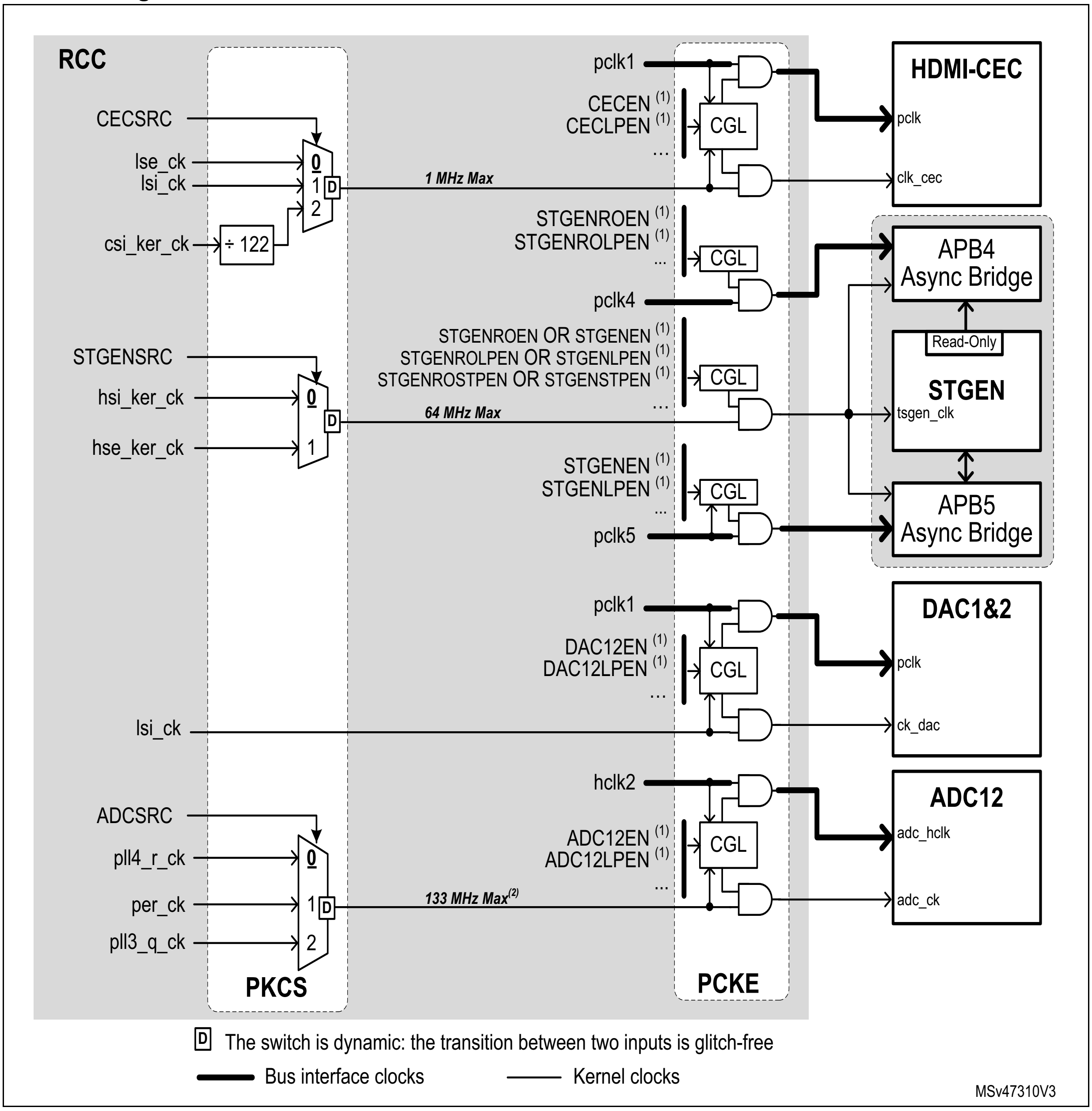
The application can select which clock can be used as STGEN clock (HSE or HSI) via the STGENSRC bit of *RCC STGEN clock selection register (RCC_STGENCKSELR)*.

In order to ensure that the clock is provided to the STGEN when the system is in (LP-)Stop mode, the application shall set the bits STGENEN, STGENLPEN and STGENSTPEN to ‘1’ or set the bits STGENROEN, STGENROLPEN and STGENROSTPEN to ‘1’. In addition, HSEKERON or HSIKERON bits must be set to ‘1’, according to the clock source selected.

The ADC12 has an embedded prescaler in order to provide to the ADC analog cell a clock with a duty cycle as close as possible to 50%.

- If the embedded ADC12 prescaler is bypassed, it is up to the RCC to provide a **clk_adc** clock with a duty cycle of 50%.
  Note that in order to get a clk_adc with a duty cycle of 50%, the post dividers (DIVQ or DIVR) must be programmed to an odd value. If hsi_ker_ck is used, HSIDIV[1:0] must be different from 0.
- If the embedded ADC12 prescaler is used, then the RCC can provide a clock at a frequency up to 133 MHz.

Figure 85. Kernel clock distribution for ADC12, DACs, STGEN and HDMI-CEC

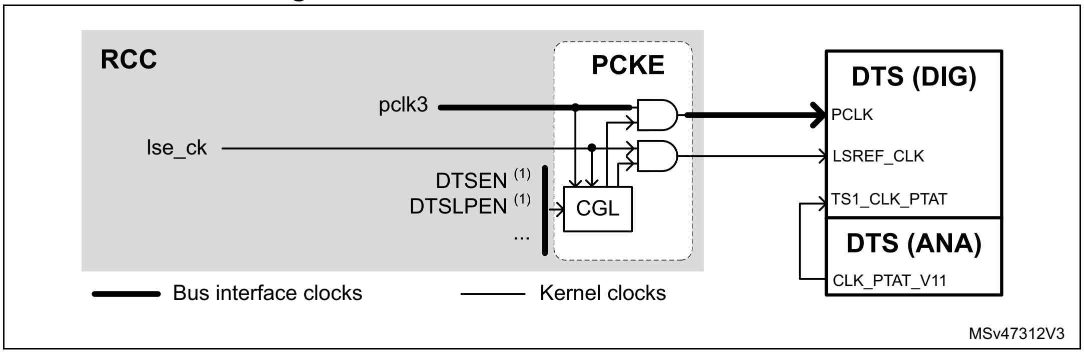


1. There is one enable bit and one low-power enable bit for MPU and MCU.
   X Represents the selected MUX input after a system reset.
2. In case ADC prescaler is bypassed, max frequency is limited by the ADC specification (see product datasheet).

Clock distribution for digital temperature sensor (DTS)

The DTS block needs an APB clock and the **lse_ck** clock. The RCC can provide the lse_ck clock to the DTS block even when the system is in (LP-)Stop. Refer to section *Section 10.4.11: Peripheral clock gating control* for additional information.

Figure 86. Kernel clock distribution for DTS



1. There is one enable bit and one low-power enable bit for MPU and MCU.

Clock distribution for RTC/AWU

The **rtc_ck** clock source can be:

- The **hse_rtc_ck** (**hse_ck** divided by a programmable prescaler),
- The **lse_ck** or,
- The **lsi_ck** clock.

The source clock is selected by programming the RTCSRC[1:0] bits in the *RCC backup domain control register (RCC_BDCR)* and the RTCDIV[5:0] bits in the *RCC RTC clock division register (RCC_RTCDIVR)*.

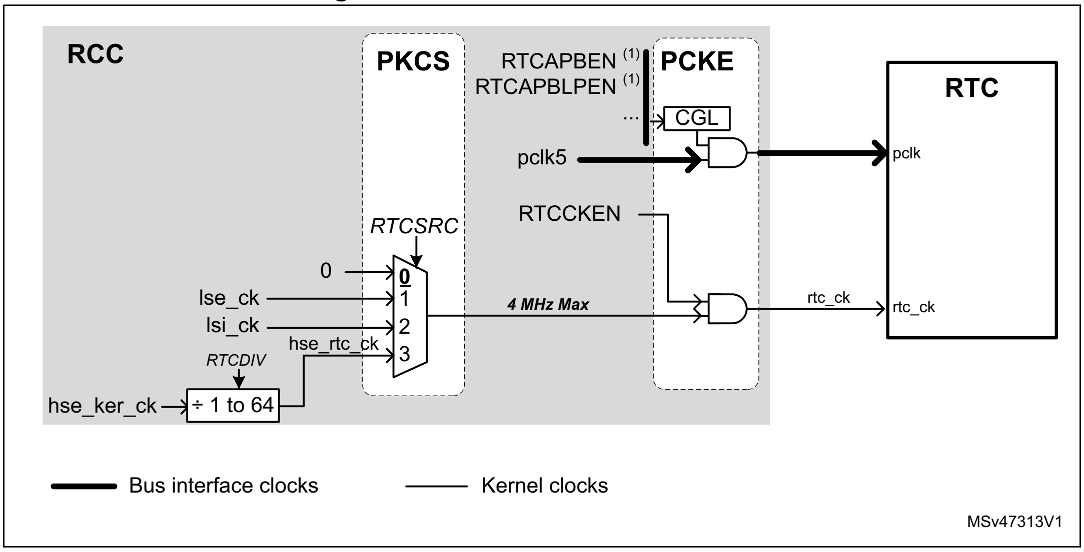
The RTCSRC[1:0] can be modified only once after a reset of the backup domain, or after a LSE failure (LSECSSD = ‘1’).

Note as well that in order to modify the RCC_BDCR register, the DBP bit in the PWR control register 1 (PWR_CR1) has to be set to ‘1’.

The LSE oscillator is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as the RTC clock:
  - The RTC continues to work even if the V_{DD} supply is switched off, provided the VBAT supply is maintained.
- If LSI is selected as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off.
  - The RTC continues to work in Stop, LP-Stop, LPLV-Stop and Standby modes.
- If the HSE clock is used as the RTC clock:
  - The RTC state is not guaranteed if the V_{DD} supply is powered off or if the V_{DDCORE} supply is powered off.
  - The RTC continues to work in (LP-)Stop mode if HSEKERON = ‘1’. Note that the clock is lost after a **nreset**.

Figure 87. Clock distribution for RTC



1. There is one enable bit and one low-power enable bit for MPU and MCU.
X Represents the selected MUX input after a system reset.

The **rtc_ck** clock is enabled via RTCCKEN bit located into the RCC_BDCR register.
The RTC bus interface clock (APB clock) is enabled via RTCAPBEN and RTCAPBLPEN bits located respectively into RCC_MP_APB5ENSETR, RCC_MC_APB5ENSETR, RCC_MP_APB5LPENSETR and RCC_MC_APB5LPENSETR registers.

The RTC bus interface clock (APB clock) is disabled via RTCAPBEN and RTCAPBLPEN bits located into RCC_MP_APB5ENCLRR, RCC_MC_APB5ENCLRR, RCC_MP_APB5LPENCLRR and RCC_MC_APB5LPENCLRR registers.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTCLCK}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result as the first one. Otherwise a third read access must be performed.

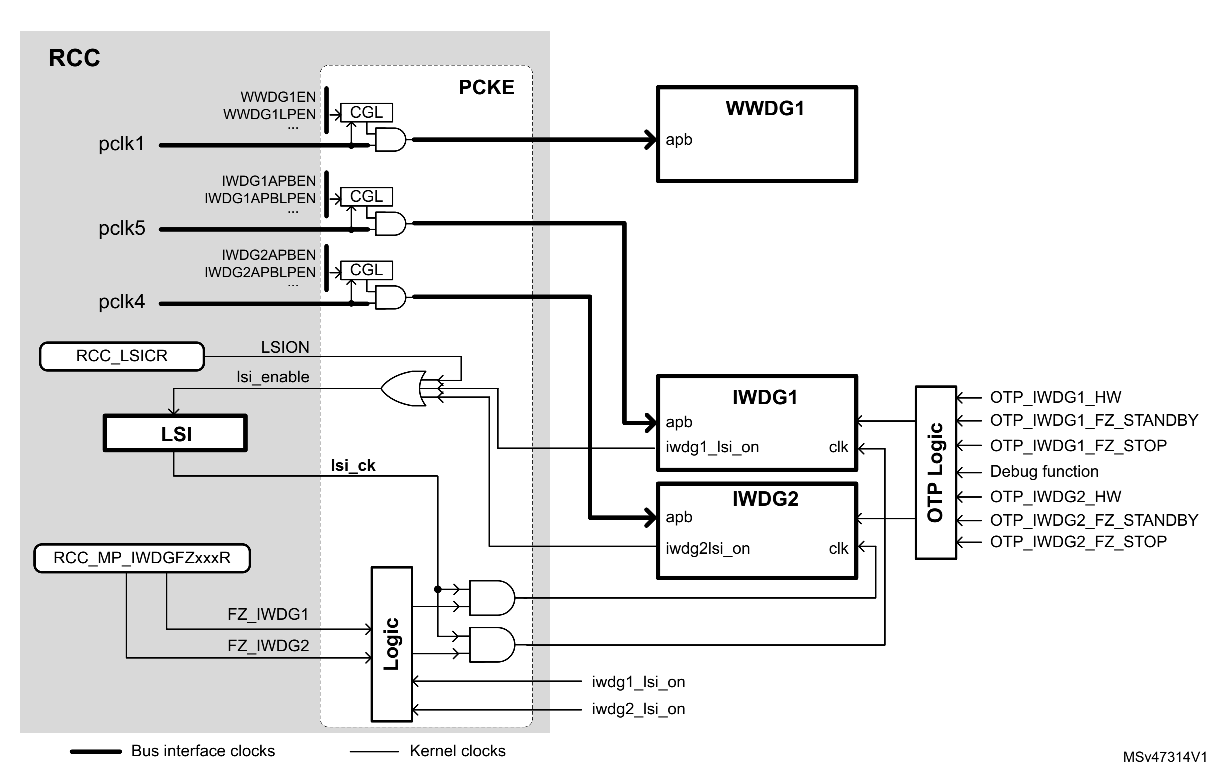
Clock distribution for watchdogs (IWDG[2:1], WWDG1)

The RCC provides the clocks for the three watchdog blocks available on the circuit:

- The two independent watchdogs (IWDG1 and IWDG2) are connected to the LSI.
- The window watchdog WWDG1 is connected to the APB1 clock.

If an independent watchdog is started by either option byte configuration or by software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator set-up delay, the clock is provided to the IWDGs.

Figure 88. Clock distribution for IWDG[2:1] and WWDG1



WWDG1 clock control

The window watchdog clock (**pclk1**) can be enabled by setting the WWDG1EN bit in RCC_MC_APB1ENSETR register. The software cannot stop WWDG1 down-counting by setting WWDG1EN bit to ‘0’. It is expected that the RCC_MC_ABP1ENSETR register is controlled by the MCU, but it is also possible for the MPU to access this register if requested by a specific application.

The WWDG1 block is frozen when the MCU goes to CStop mode.

IWDG[2:1] clock control

The independent watchdogs do not need their clock to be enabled by the RCC block. There are two ways to provide the LSI clock to the IWDG1/2:

- Enabling the IWDG1/2 by software, and use the sequences described into *Section 46.4.4: Window option*. The IWDG1/2 will request automatically the LSI clock activation to the RCC.
- Via the OTP_IWDG[2:1]_HW option bytes, which allow IWDG1 and IWDG2 to be automatically enabled after a system reset. Refer to *Section 46.4.4: Window option* configuration for watchdog for details.

Note as well that the IWDG1 and IWDG2 bus interface clocks (APB clock) must be enabled before accessing the IWDGx registers. The APB clock can be disabled when no access to IWDGx is required. The IWDGx are still counting-down even if the APB clock is disabled.

Note as well that the IWDG1 bus interface clock (APB clock) is controlled via IWDG1APBEN and IWDG1APBLPEN bits located into RCC_MP_APB5ENSETR, RCC_MP_APB5LPENSETR, RCC_MP_APB5ENCLRR and RCC_MP_APB5LPENCLRR registers.

In the same way, the IWDG2 bus interface clock (APB clock) is controlled via IWDG2APBEN and IWDG2APBLPEN bits located into RCC_MP_APB4ENSETR, RCC_MP_APB4LPENSETR, RCC_MP_APB4ENCLRR and RCC_MP_APB4LPENCLRR registers for the MPU.

The LSI clock at the IWDG[2:1] inputs can be frozen in the following situations:

- When the bits FZ_IWDG[2:1] are set (see *Handling of the IWDGs during BOOTROM code execution*). These bits are dedicated to the BOOTROM, and are used to freeze the IWDG[2:1] during BOOTROM code execution.
- When the option bytes OTP_IWDG[2:1]_FZ_STANDBY are set. In this case the corresponding IWDG[2:1] is frozen when the system goes to Standby.
- When the option bytes OTP_IWDG[2:1]_FZ_STOP are set. In this case the IWDG[2:1] are frozen when the MPU goes CStop.

Note that the IWDG[2:1] can also be frozen in debug mode.

Handling of the IWDGs during BOOTROM code execution

The BOOTROM code is always activated after a system reset, a Standby, or a CStandby reset.

As soon as the BOOTROM code is activated, the BOOTROM will freeze the IWDGs clocks. This is done by setting the FZ_IWDG1 and FZ_IWDG2 bits of *RCC IWDG clock freeze set register (RCC_MP_IWDGFZSETR)* to ‘1’.
Before giving the control to the application, the BOOTROM writes the FZ_IWDG1 and FZ_IWDG2 bits of *RCC IWDG clock freeze clear register (RCC_MP_IWDGFZCLRR)* to ‘1’ in order to unfreeze the IWDGs. In that way if the IWDGs were enabled by the application, they will be re-enabled, otherwise they remain disabled.

The access to RCC_MP_IWDGFZSETR and RCC_MP_IWDGFZCLRR registers are restricted to the MPU, and only in secure mode.

The RCC_MP_IWDGFZSETR register used to freeze the IWDGs, is very sensitive. Every time the MPU is reset (i.e. system reset: **nreset or** CStandby reset: **cstby_rst**), the hardware allows to write the RCC_MP_IWDGFZSETR register only once.

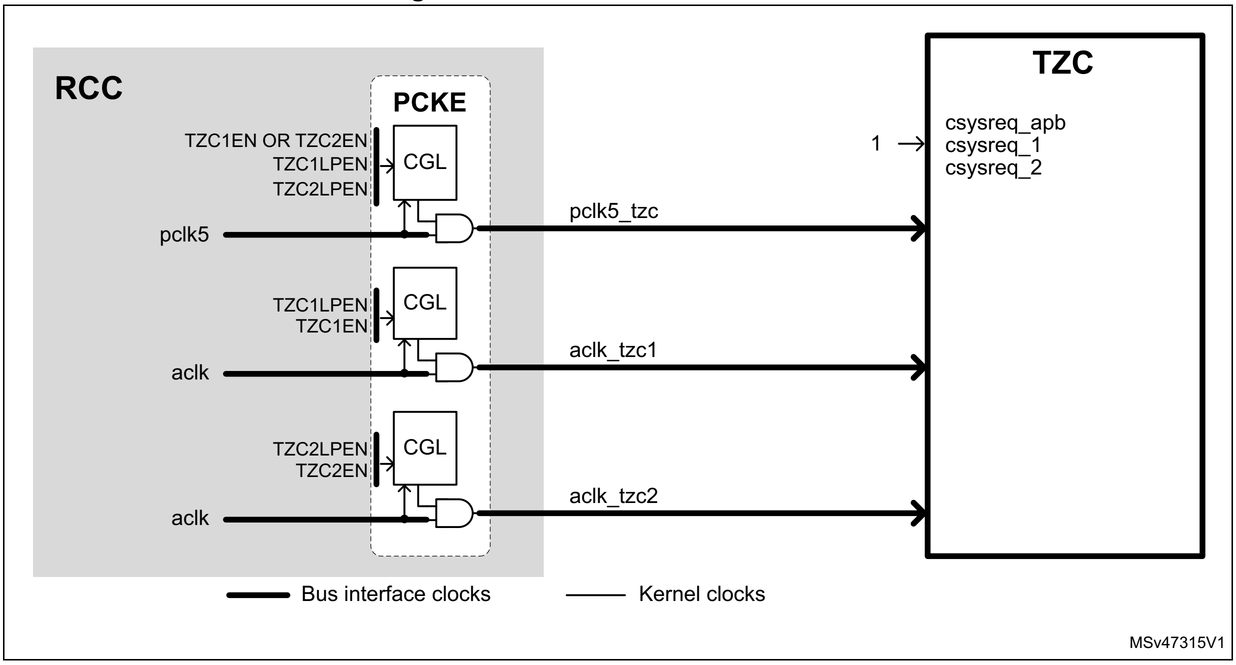
Clock distribution for TZC

The next figure shows a simplified view of the clock distribution for the TZC. The TZC receives the following clocks:

- An APB clock (**pclk5_tzc**),
- An AXI clock for AXI port 1 (**aclk_tzc1**),
- An AXI clock for AXI port 2 (**aclk_tzc2**).

The application can choose to enable the APB clock and one or both clocks of the AXI ports.

Figure 89. Clock distribution for TZC



1. There is one enable bit and one low-power enable bit for MPU and MCU.

In order to enable the TZC, the processor shall set TZC1EN and/or TZC2EN to ‘1’. If the application wants to keep the clocks active in CSleep, the corresponding bit TZC1LPEN and/or TZC2LPEN must be set to ‘1’ as well.

Each processor (MPU and MCU) have their own enable bit control on RCC_Mx_APB5ENSETR/CLRR and RCC_Mx_APB5LPENSETR/CLRR registers.

Clock distribution for DDRC and DDRPHYC

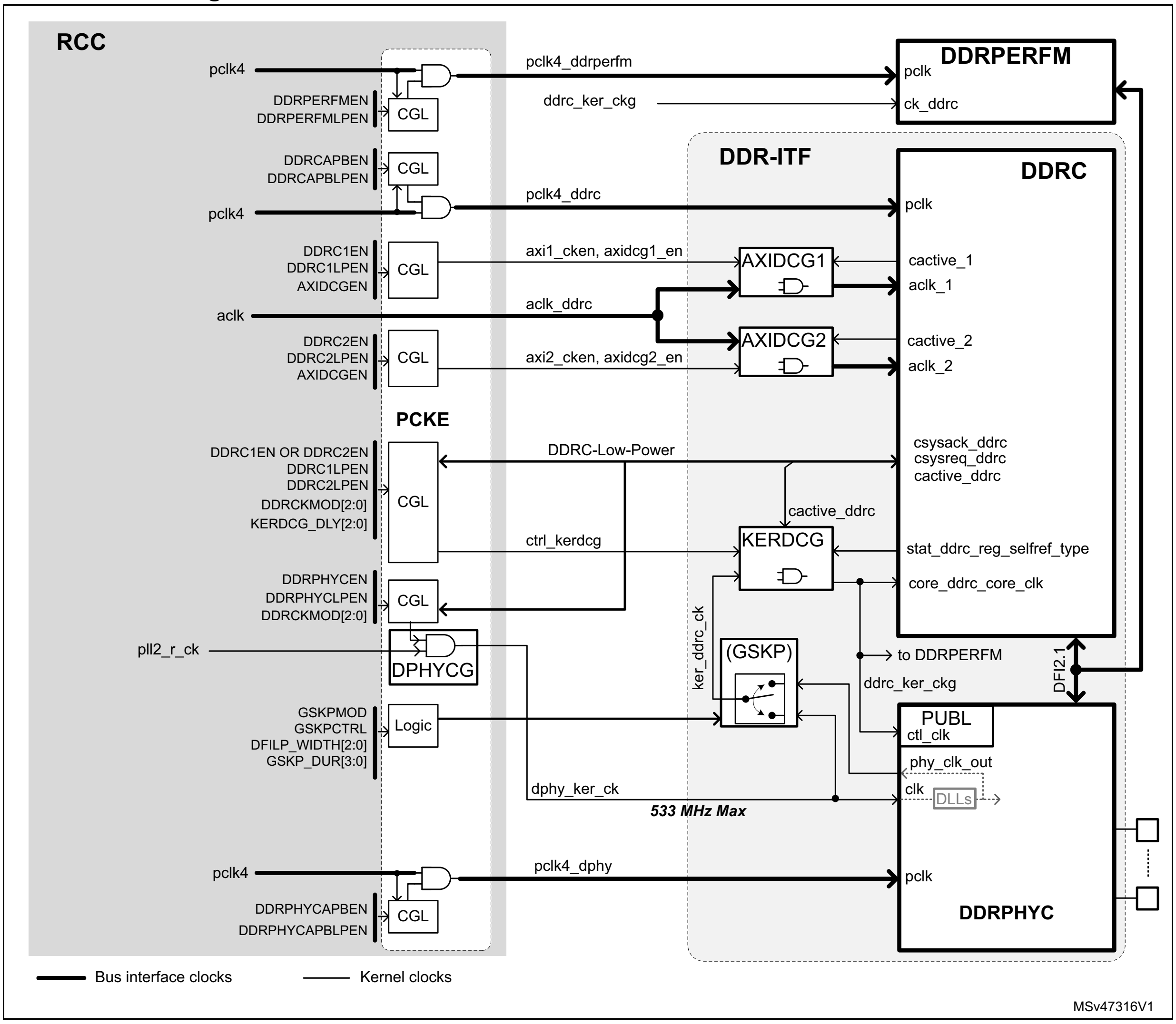
The next figure shows a simplified view of the clock distribution for the DDRC and DDRPHYC blocks. Refer to *Section 5: DDR3/LPDDR2/LPDDR3 controller (DDRCTRL)* for additional information.

The RCC needs to provide the following clocks:

- The APB clocks for the register interface (**pclk4_ddrperfm, pclk4_ddrc** and **pclk4_dphy**).
- The AXI clocks for the two AXI ports of DDRC (**aclk**).
- The kernel clock for the DDRPHYC (**dphy_ker_ck**). Note that the DDRPHYC provides the clock **ddrc_ker_ck** for the DDRC, DDRPERFM and PUBL (part of the DDRPHYC).

Note that all RCC control bits mentioned in this section are located into to the *RCC DDR interface control register (RCC_DDRITFCR)*.

Figure 90. Clock distribution for DDRC, DDRPHYC and DDRPERFM



Control of the APB clocks

The control of the APB clock gating is performed via dedicated bits:

- The DDRCAPBEN and DDRCAPBLPEN bits control the gating of the APB clock of the DDRC interface (**pclk4_ddrc**),
- The DDRPHYCAPBEN and DDRPHYCAPBLPEN bits control the gating of the APB clock of the DDRPHYC interface (**pclk4_dphy**),

Refer to *Section 10.4.11: Peripheral clock gating control* for details.

Control of the AXI clocks

The AXIDCG[2:1] blocks are handling the gating of the AXI clocks (**aclk_[2:1]**) provided to the DDRC AXI ports. The AXIDCG[2:1] bits are controlled by the RCC, and can work as follow:

- Automatic mode: the **aclk_[2:1]** clocks are gated according to **cactive_[2:1]** signals provided by the DDRC.
- Manual mode: the RCC can directly enable or disable the AXI clock generation.

The application can disable the clock of one AXI port, when not used, by setting the corresponding DDRCxEN to ‘0’.

Table 59. Gating control of the aclk_[2:1] clocks

DDRC1EN (DDRC2EN)	DDRC1LPEN (DDRC2LPEN)	AXIDCGEN	MPU mode	-	aclk_1 (aclk_2) clock	Comments
0	X	X	Any	→	OFF	The **aclk_x** is gated because DDRCxEN = ‘0’.
1	X	0	CRun	→	ON	The **aclk_x** is forced ON independently of **cactive_x** because DDRCxEN = ‘1’, AXIDCGEN = ‘0’ and MPU is in CRun
		1		→	DCG (1)	When DDRCxEN = AXIDCGEN = ‘1’, the **aclk_x** is gated automatically according to **cactive_x**.
1	0	0	CSleep	→	OFF	The **aclk_x** is gated because DDRCxLPEN = ‘0’, and AXIDCGEN = ‘0’, and the MPU is in CSleep
	1			→	ON	The **aclk_x** is forced ON independently of **cactive_x** because DDRCxEN = DDRCxLPEN = ‘1’ and AXIDCGEN = ‘0’
	X	1		→	DCG	When DDRCxEN = AXIDCGEN = ‘1’, the **aclk_x** is gated automatically according to **cactive_x**.
X	X	X	CStop or CStandby	→	OFF	The **aclk_[2:1]** are gated.

1. **DCG** means Dynamic Clock Gating according to **cactive_x** signal, highlighted in gray.

Control of the DDRC and DDRPHYC kernel clocks

The KERDCG and DPHYCG blocks are handling respectively the clock gating of the **ddrc_ker_ckg** and **dphy_ker_ck.** They are controlled by the RCC, and can work in normal, automatic or hardware low-power self-refresh mode according to the way the DDRC and the RCC are programmed.

Table 60 details the way clocks **ddrc_ker_ckg** and **dphy_ker_ck** are handled according to the value of DDRCKMOD.

Table 60. DDRCKMOD control description

<table>
<tr><th>DDRCKMOD</th><th>-</th><th>KERDCG mode</th><th>DPHYCG mode</th><th>Comment</th></tr>
<tr><td>‘000’</td><td>→</td><td>Normal</td><td>Normal</td><td>Software Self-Refresh mode (SSR): (1)
The gating of the dphy_ker_ck clock depends on the DDRPHYCEN, DDRPHYCLPEN and MPU mode.
The gating of the ddrc_ker_ckg clock depends on the DDRCxEN, DDRCxLPEN and MPU mode.</td></tr>
<tr><td>‘001’</td><td>→</td><td rowspan="2">Auto.</td><td>Normal</td><td>Automatic Self-Refresh mode (ASR1): (2) (3)
The clock ddrc_ker_ckg is gated automatically according to cactive_ddrc signal.
The gating of the dphy_ker_ck clock depends on the DDRPHYCEN, DDRPHYCLPEN and MPU mode.
DDRCxEN is still used to disable/enable the DDR interface,</td></tr>
<tr><td>‘101’</td><td>→</td><td>Auto</td><td>Full Automatic Self-Refresh mode (ASR2): (4) (3)
The clocks ddrc_ker_ckg and dphy_ker_ck are gated automatically according to cactive_ddrc signal.
In this mode, the bit DDRPHYCEN must be set to ‘1’, and DDRPHYCLPEN must be set to ‘0’.
DDRCxEN is still used to disable/enable the DDR interface,</td></tr>
<tr><td>‘010’</td><td>→</td><td rowspan="2">DDRC-LP</td><td>Normal</td><td>Hardware Self-Refresh mode (HSR1): (2) (5)
The gating of the ddrc_ker_ckg clock is controlled by the DDRC Low-Power interface connected to the DDRC.
The gating of the dphy_ker_ck clock depends on the DDRPHYCEN, DDRPHYCLPEN and MPU mode.
DDRCxEN is still used to disable/enable the DDR interface.</td></tr>
<tr><td>‘110’</td><td>→</td><td>DDRC-LP</td><td>Full Hardware Self-Refresh mode (HSR2): (4) (5)
The gating of ddrc_ker_ckg and dphy_ker_ck clocks are controlled by the DDRC Low-Power interface connected to the DDRC.
In this mode, the bit DDRPHYCEN must be set to ‘1’, and DDRPHYCLPEN must be set to ‘0’.
DDRCxEN is still used to disable/enable the DDR interface.</td></tr>
</table>

1. This mode must be selected during DDRC and DDRPHYC initialization phase, and if the application is using the software self-refresh.
2. This mode can be used when the application wants to maintain a clock to the DLLs.
3. It is recommended to have DDRCxEN = DDRCxLPEN = DDRPHYCEN = DDRPHYCLPEN = ‘1’.
4. This mode can be used when the DLLs are in bypass.
5. It is recommended to have DDRCxEN = DDRPHYCEN = ‘1’ and DDRCxLPEN = DDRPHYCLPEN = ‘0’.

Glitch skipper (GSKP)

The RCC also needs to handle a block named Glitch Skipper (GSKP). The GSKP is performing following functions:

- The GSKP prevents low-power mode transitions shorter than 1 us. The application has to program a timeout delay via the DFILP_WIDTH[2:0] field.
- The GSKP prevents to have glitches on the clock **ddrc_ker_ck** during the re-activation of the DLLs. For that purpose, when GSKPMOD = ‘1’, the GSKP is providing the clock **dphy_ker_ck** to the DDRC for a duration given by the GSKP_DUR[3:0] field.

  The application can also control manually the clock switching via GSKPCTRL bit when GSKPMOD = ‘0’. In this case, when GSKPCTRL is set to ‘1’, the GSKP provides the clock **dphy_ker_ck** to the DDRC, and when GSKPCTRL is set to ‘0’, the GSKP will provide the clock **phy_clk_out** to the DDRC, after a delay given by GSKP_DUR[3:0].

  The DFILP_WIDTH, GSKPMOD, GSKP_DUR and GSKPCTRL fields are located in *RCC DDR interface control register (RCC_DDRITFCR)*.

DDRC Low-power Interface

The DDRC block provides a low-power interface which can used to control the entry and exit of the DDRC and DDRPHYC into low-power mode. This interface also controls the gating of the **dphy_ker_ck** and **ddrc_ker_ckg** clocks. This low-power interface is compliant to AMBA AXI protocol specification. In order to simplify the drawings, this document only shows the most significant states of the DDRC low-power interface. *Table 60* gives details on the DDRC low-power states.

The DDRC Low-power interface can be used in HSR1 and HSR2 modes in order to control:

- The generation of the DDRC and DDRPHYC kernels clocks,
- The entry/exit of the external DDR device into low-power (i.e. self-refresh).

Table 61. DDRC low-power (DDRC-LP) states description

DDRC low-power states	-	Comment
CKON	→	Normal clocked operation: The RCC provides the kernel clock(s) to the DDRC and DDRPHYC. **csysreq_ddrc**, **csysack_ddrc** and **cactive_ddrc** are all set to ‘1’
CKOFF	→	Low-power operation: The RCC does not provide the kernel clock(s) to the DDRC and DDRPHYC. **csysreq_ddrc**, **csysack_ddrc** and **cactive_ddrc** are all set to ‘0’
CKON_Rreq	→	Low-power exit initiated by the RCC: The RCC provides the kernel clock(s) to the DDRC and DDRPHYC. The RCC sets the **csysreq_ddrc** to ‘1’, and waits for **csysack_ddrc** and **cactive_ddrc** set to ‘1’ before going to state CKON.
CKON_Preq	→	Low-power exit initiated by the DDRC: The DDRC requests the kernel clock(s) by setting the **cactive_ddrc** to ‘1’. The RCC provides the kernel clock(s), sets the **csysreq_ddrc** to ‘1’, and waits for **csysack_ddrc** set to ‘1’ before going to state CKON.
CKOFF_Rreq	→	Low-power entry initiated by the RCC: The RCC requests to the DDRC to entry into low-power by setting the **csysreq_ddrc** to ‘0’. The DDRC may accept or deny the request. If a request is denied, the interface come back to CKON state, and the RCC retries a new low-power request entry. When the low-power request is accepted by the DDRC (**csysack_ddrc** and **cactive_ddrc** = ‘0’), the RCC disables the kernel clock(s). Then the states switches to CKOFF.

Self-refresh mode control

Switching the external DDR device into self-refresh is important for energy saving.

When the external DDR device is in self-refresh, the application can switch-off the clocks of the DDRC and DDRPHYC, without loosing the content of the external DDR.

When the external DDR device is in self-refresh, the application can no longer access to the content of the external DDR device.

Exiting a DDR device from self-refresh can take several microseconds and the application has to take into account this latency. This latency is due to the external DDR device, but also to the way the DLLs located into the DDRPHYC are handled.

The RCC offers 3 ways to control the clocks of the DDRC and DDRPHYC:

- The software self-refresh mode (SSR),
- The automatic self-refresh mode (ASR),
- The hardware self-refresh mode (HSR).

The behavior of the RCC is different according to the self-refresh mode selected by the application. For that purpose, the application has to program the DDRCKMOD in accordance with the configuration of the DDRC and DDRPHYC.

The software self-refresh mode (SSR)

In this mode, the application has to program directly the DDRC in order to switch the external DDR device immediately into self-refresh. Typically when the MPU wants to go to CSleep or CStop, it must ensure that the DDR resource is no longer needed, and program the DDRC in order to switch it to self-refresh.
The piece of code performing the entry and the exit of the DDRC to self-refresh must be located into the SYSRAM. This is due to the fact that some software routine need to run when the DDR device is in self-refresh.

The automatic self-refresh mode (ASR1 and ASR2)

In this mode the software does not need to program the DDRC every-time the DDRC must be set into self-refresh. The DDRC will go automatically into self-refresh mode when the DDRC is inactive for an amount of time defined into the DDRC.

Even when the MPU is in CRun mode, the DDRC may go to self-refresh if there is no activity on its AXI ports. The automatic self-refresh mode, allows the entry into self-refresh independently of the processor mode.

When the MPU goes to CSleep, if the clocks of the DDRC and optionally the clock of the DDRPHYC need to be gated, the RCC delays the gating of these clocks until the DDRC switches into self-refresh mode.

If the system needs to go to one of the Stop modes, the RCC will delay the gating of the clocks until the DDRC switches into self-refresh mode.

The ASRx modes must be used only if the DDRC has been programmed in automatic self-refresh mode, otherwise the behavior of the DDR interface is not guaranteed.

The ASRx modes only impact the gating of the DDRC and DDRPHYC kernel clocks (i.e. **ddrc_ker_ckg** and **dphy_ker_ck**).

Warning: The ASR2 mode can only be used when the DLLs of the DDRPHYC are bypassed. So only DDR devices such as LPDDR2, working at frequencies around 125 MHz can use this feature.

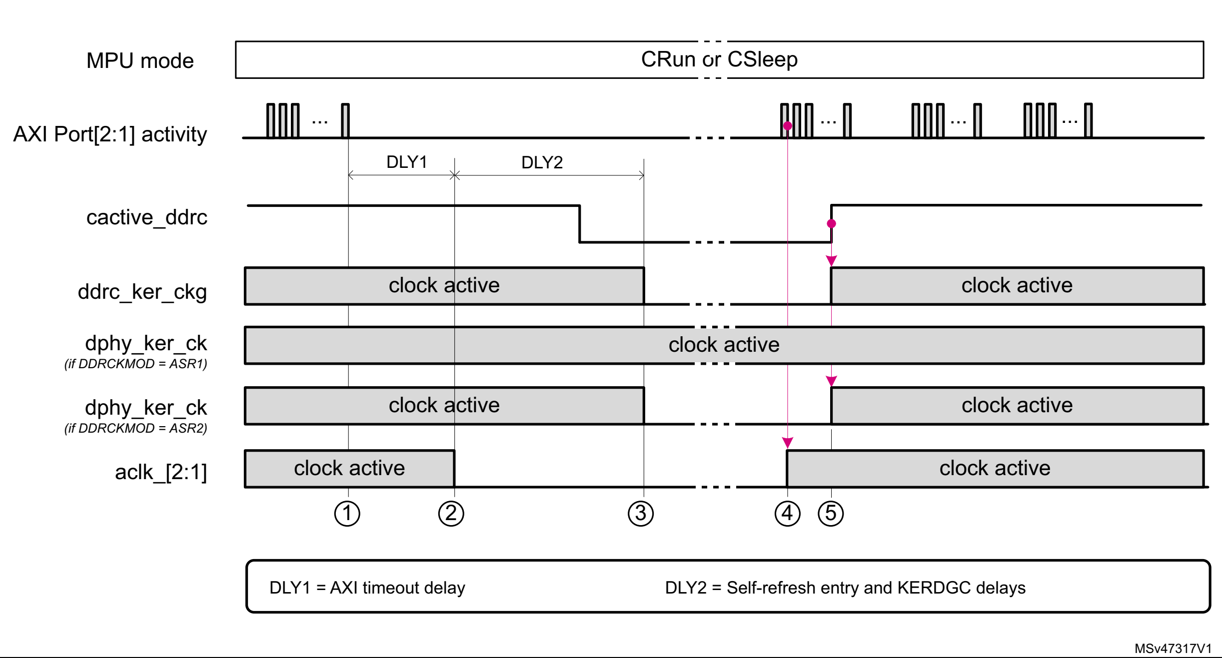
When the ASRx modes are used, at least one of the DDRCxEN bits must be set to ‘1’.

Figure 91 shows a simplified timing for the entry and exit from self-refresh.

In this example DDRCxEN = DDRCxLPEN = ‘1’, DDRPHYCEN = DDRPHYCLPEN = ‘1’, AXIDCGEN = ‘1’.

1. The activity on AXI port x is stopped, due for example, to the fact that the MPU goes to CSleep, but it could be also the case between two activity bursts.
2. When no activity is detected in the AXI port x for at least UMCTL2_AXI_LOWPWR_NOPXCNT ***aclk_x** cycles (DLY1), the AXI clock (**aclk_x**) is gated if AXIDCGEN = ‘1’.
3. When the AXI interfaces are gated, if it does not switch back to active after a delay DLY2, the clock **ddrc_ker_ckg** is gated. If the ASR1 mode is selected, the clock **dphy_ker_ck** remains enabled if DDRPHYCEN = DDRPHYCLPEN = ‘1’. If the ASR2 mode is selected, then the clock **dphy_ker_ck** is gated just after the gating of the **ddrc_ker_ckg** clock. The MPU mode can be CRun or CSleep.
4. If a transaction is performed in one of the two AXI ports, the AXI clock (**aclk_x**) is re-enabled. The AXI transaction can be caused by any master using DDR resources.
5. Few AXI clock cycles later, the DDRC sets the **cactive_ddrc** to ‘1’ in order to request the kernel clock. The DDRPHYC clock **dphy_ker_ck** and the DDRC clock **ddrc_ker_ckg** are re-enabled.The external DDR is then ready to process the pending transactions.

Figure 91. Entering and exiting self-refresh in ASR1 or ASR2 modes



The hardware self-refresh mode (HSR1 and HSR2)

The hardware self-refresh mode is using the DDRC-low-power interface (DDRC-LP) in order to request to the DDR to go or exit from low-power.

The DDRC-LP allows the exit of the DDR interface from self-refresh in the following cases:

- When the interface is enabled via DDRCxEN bits,
- When the MPU exits from CSleep or CStop,
- When the DDRC requests to exit from self-refresh. This could happen if another master is accessing one of the AXI ports.

The DDRC-LP allows the entry of the DDR interface into self-refresh in the following cases:

- When the interface is disabled via DDRCxEN bits,
- When the MPU transitions from CRun to CSleep,
- When the MPU transitions from CRun to CStop.

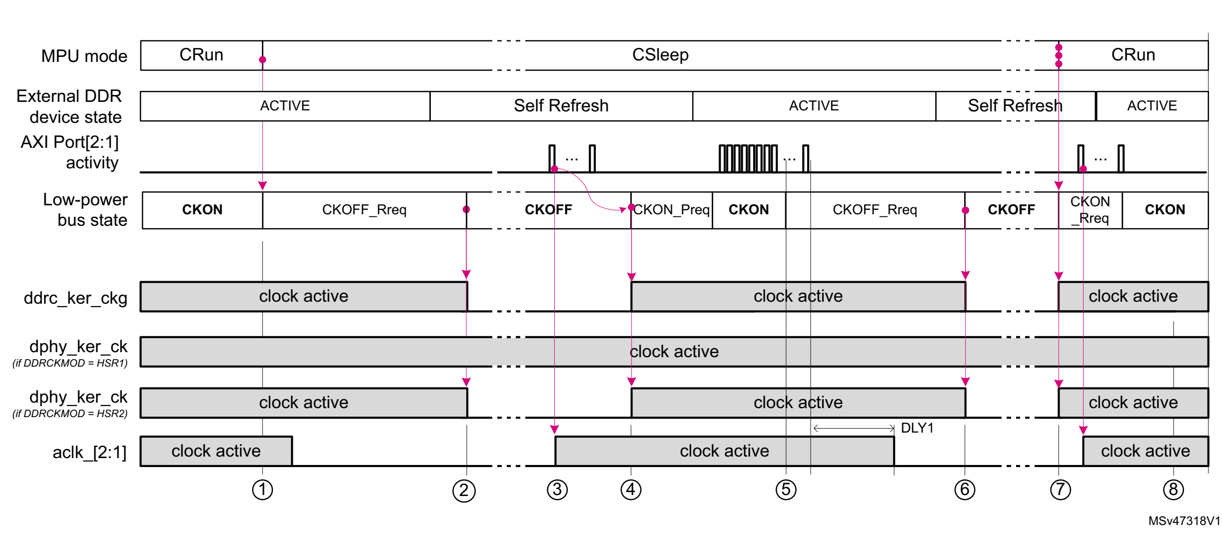
This may be less efficient in term of power saving than the automatic mode (ASR) but avoids that the DDRC goes accidentally to self-refresh when the MPU is in CRun, increasing the system performances.

Warning: The HSR2 mode can only be used when the DLLs of the DDRPHYC are bypassed. So only DDR devices such as LPDDR2, working at frequencies around 125 MHz can use this feature.

Figure 92 shows simplified timing for the self-refresh entry and exit. This is typically what will happen when the MPU goes to CSleep while some master are still enabled.

This example is based on configurations #7 and #8 of *Table 61*, with AXIDCGEN = ‘1’.

Figure 92. Entering and exiting self-refresh in HSR1 or HSR2 modes



1. When the MPU goes to CSleep, the RCC requests to the DDRC to go to self-refresh, using the DDRC Low-power bus (CKOFF_Rreq). Note as well that the AXI clock will be disabled if no transactions occur on AXI ports for a programmed delay (DLY1).
2. When all the conditions in the DDRC are met (no transaction pending, DDR in self-refresh), the DDRC acknowledges the request. Note that if the DDRC denies the request from the RCC, the RCC has to generate a new requests until it is accepted by the DDRC, or if the MPU exits from CSleep.
   When the DDRC acknowledges the low-power request, then the RCC disables the DDRC clock **ddrc_ker_ckg**. If the HSR1 mode is selected, the clock **dphy_ker_ck** remains enabled if DDRPHYCEN = DDRPHYCLPEN = ‘1’. If the HSR2 mode is selected the DDRPHYC clock **dphy_ker_ck** is disabled as well. The DDRC low-power bus is then in CKOFF state.
3. If a transaction is performed in one of the two AXI ports, the AXI clock (**aclk_x**) is re-enabled. The AXI transaction can be caused by any master using DDR resources.
4. Few AXI clock cycles later, the DDRC request to the RCC to re-enable the kernel clock, using the DDRC Low-power bus (CKON_Preq). As soon as the RCC receives the request, the DDRPHYC clock **dphy_ker_ck** and the DDRC clock **ddrc_ker_ckg** are re-enabled. The external DDR is then ready to process the pending transactions.
5. As the MPU is still in CSleep mode, the RCC will request to the DDRC to go to self-refresh, using the DDRC Low-power bus (CKOFF_Rreq). The DDRC may deny the request several times if all conditions to enter in self-refresh are not met. The RCC has to renew its request regularly until it is accepted by the DDRC. The RCC shall stop performing this request if the MPU goes to CRun meanwhile.
6. When all the conditions in the DDRC are met (no transaction pending, DDR in self-refresh), the DDRC acknowledges the request. When the DDRC acknowledges the low-power request, then the RCC disables the DDRC clock **ddrc_ker_ckg**. If the HSR1 mode is selected, the clock **dphy_ker_ck** remains enabled if DDRPHYCEN = DDRPHYCLPEN = ‘1’. If the HSR2 mode is selected the DDRPHYC clock **dphy_ker_ck** is disabled as well. The DDRC low-power bus is then in CKOFF state.

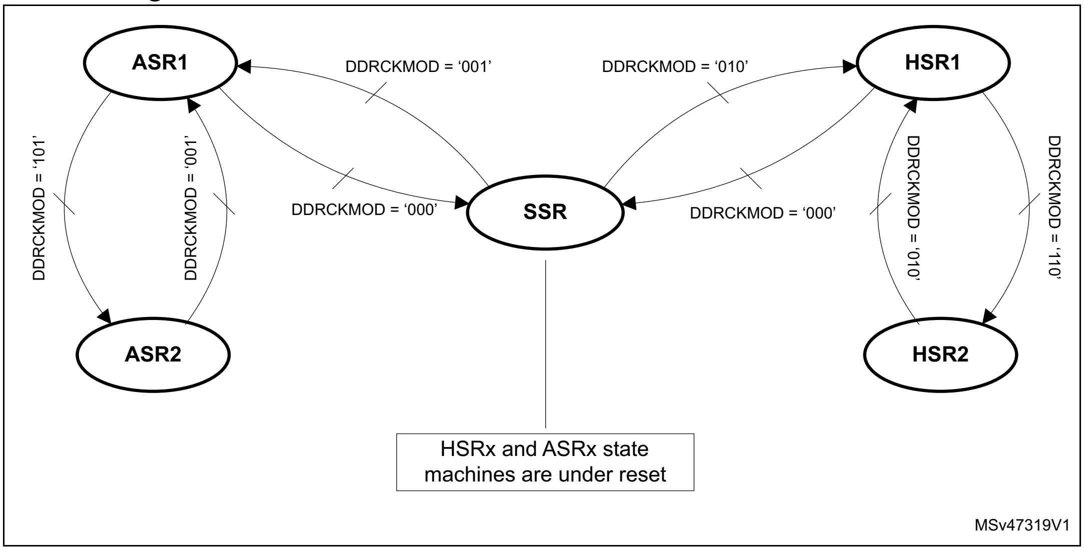
The sequence described between step 4 to step 7 may be repeated several times will the MPU is in CSleep.

7. When the MPU exits from CSleep, the RCC re-enables the DDRPHYC clock **dphy_ker_ck** and the DDRC clock **ddrc_ker_ckg**, and requests to the DDRC to exit the DDR from self-refresh, using the DDRC Low-power bus (CKON_Rreq).
8. The DDRC acknowledges the request by setting the DDRC low-power bus into CKON state. The external DDR is then ready to process the pending transactions.

Transitions between SSR, ASR and HSR

It is possible for the application to change the value of the DDRCKMOD without disabling the DDR interface. The next figure shows the allowed transition between SSR, ASR and HSR modes.

Figure 93. Allowed transitions between SSR, ASR and HSR modes



It is not allowed to transition directly:

- From ASRx to HSRx, the application needs first to set DDRCKMOD to ‘000’ (SSR), and then set the DDRCKMOD to the wanted HSR or ASR mode,
- From SSR to ASR2, the application needs first to set DDRCKMOD to ‘001’ (ASR1), and then set the DDRCKMOD to ‘101’ (ASR2 mode),
- From ASR2 to SSR, the application needs first to set DDRCKMOD to ‘001’ (ASR1), and then set the DDRCKMOD to ‘000’ (SSR mode),
- From SSR to HSR2, the application needs first to set DDRCKMOD to ‘010’ (HSR1), and then set the DDRCKMOD to ‘110’ (HSR2 mode),
- From HSR2 to SSR, the application needs first to set DDRCKMOD to ‘010’ (HSR1), and then set the DDRCKMOD to ‘000’ (SSR mode).

Recommended configurations

The bits DDRCxEN and DDRPHYC are used to enable or disable the DDR interface.

Table 62 shows the recommended RCC configurations. The application has to select the configuration adapted to its DDR configuration.

Table 62. Recommended configurations (1)

Conf #	DDRCKMOD	DDRCxEN	DDRCxLPEN	DDRPHYCEN	DDRPHYCLPEN	-	MPU CRun mode AXI clock (2)	MPU CRun mode DDRC clock	MPU CRun mode DDRPHYC clock	MPU CSleep mode AXI clock (2)	MPU CSleep mode DDRC clock	MPU CSleep mode DDRPHYC clock	Comments
1	Any	0	x	0	x	→	OFF	OFF	OFF	OFF	OFF	OFF	Interface disabled
2	SSR	1	0	1	0	→	ON	ON	ON	OFF	OFF	OFF	The gating of the DDRC and DDRPHYC clock is dependent of the MPU mode. It is up to the application software to switch the DDR interface into low-power mode, before gating the clocks.
3		1	0	1	1	→	ON	ON	ON	OFF	OFF	ON	
4		1	1	1	1	→	ON	ON	ON	ON	ON	ON	
5	ASR1	1	1	1	1	→	ON	DCG	ON	ON	DCG	ON	Only DDRC kernel clock is controlled according to automatic mode.
6	ASR2	1	1	1	0	→	ON	DCG	DCG	ON	DCG	DCG	Both DDRC and DDRPHYC kernel clocks are controlled according to automatic mode. This mode can only be used when the DLLs of the DDRPHYC are bypassed. So only DDR devices such as LPDDR2, working at frequencies around 125 MHz can use this feature.
7	HSR1	1	0	1	1	→	ON	DLP	ON	OFF	DLP	ON	The DDRC kernel clock is controlled according to DDRC-LP interface. The DDRPHYC kernel clock shall be maintained active by setting DDRPHYCEN = DDRPHYCLPEN = ‘1’
8	HSR2	1	0	1	0	→	ON	DLP	DLP	OFF	DLP	DLP	Both DDRC and DDRPHYC kernel clocks are controlled according to DDRC-LP interface. This mode can only be used when the DLLs of the DDRPHYC are bypassed. So only DDR devices such as LPDDR2, working at frequencies around 125 MHz can use this feature.

1. DCG means that the clocks are controller using dynamic clock gating, highlighted in gray.
In the case of DDRC and DDRPHYC, the clocks are gated by the **cactive_ddrc** signal.
DLP means DDRC Low-Power protocol, highlighted in blue.
2. When the bit AXIDCGEN = ‘1’, the AXI clock is gated automatically according to **cactive_[2:1]** (DCG), except when DDRCxEN = ‘0’.

In the configurations #2 to #4, the gating of the DDRC and DDRPHYC clock is dependent of the MPU mode. It is up to the application software to switch the DDR interface into low-power mode, before gating the clocks.

Configuration #2

- The APB and the kernel clocks of both DDRC and DDRPHYC are enabled when the MPU is in CRun, and disabled in CSleep.

Configuration #3

- The APB clock of DDRC is enabled when the MPU is in CRun, and disabled in CSleep.
- The APB clock of DDRPHYC is enabled when the MPU is in CRun or CSleep.
- The DDRC kernel clock (**ddrc_ker_ckg**) is enabled when the MPU is in CRun, and disabled in CSleep.
- The DDRPHYC kernel clock (**dphy_ker_ck**) is enabled when the MPU is in CRun or CSleep.

Configuration #4

- The APB clocks of DDRC and DDRPHYC are enabled when the MPU is in CRun or CSleep.
- The DDRC and DDRPHYC kernel clocks (**ddrc_ker_ckg** and **dphy_ker_ck**) are enabled when the MPU is in CRun or CSleep.

In the configurations #5 and #6, the gating of the DDRC and DDRPHYC clocks is automatic and dependent to the activity of the DDRC.

Configuration #5

- The APB clock of both DDRC and DDRPHYC are enabled when the MPU is in CRun or CSleep.
- The DDRC kernel clock (**ddrc_ker_ckg**) is gated according to **cactive_ddrc** independently of the MPU mode.
- The DDRPHYC kernel clock (**dphy_ker_ck**) is enabled independently of the MPU mode. This configuration can be interesting when the DLLs of the DDRPHYC need to remain locked.

Configuration #6

- The APB clock of both DDRC and DDRPHYC are enabled when the MPU is in CRun or CSleep.
- The DDRC and DDRPHYC kernel clocks (**ddrc_ker_ckg** and **dphy_ker_ck**) are gated according to **cactive_ddrc** independently of the MPU mode.
- This configuration saves power when the DLLs of the DDRPHYC are not used (bypassed).

In the configurations #7 to #8, the gating of the DDRC and DDRPHYC clocks is controlled by the DDRC-LP. The DDRC-LP can also request to the DDR interface to go to low-power (i.e. self-refresh).

Configuration #7

- The APB clock of DDRC is enabled when the MPU is in CRun, and disabled in CSleep.
- The APB clock of DDRPHYC are enabled when the MPU is in CRun or CSleep.
- The DDRC kernel clock (**ddrc_ker_ckg**) is gated using to the DDRC-Low-power handshake, according to the MPU mode, and DDRC requests.
- The DDRPHYC kernel clock (**dphy_ker_ck**) is enabled independently of the MPU mode. This configuration can be interesting when the DLLs of the DDRPHYC need to remain locked.

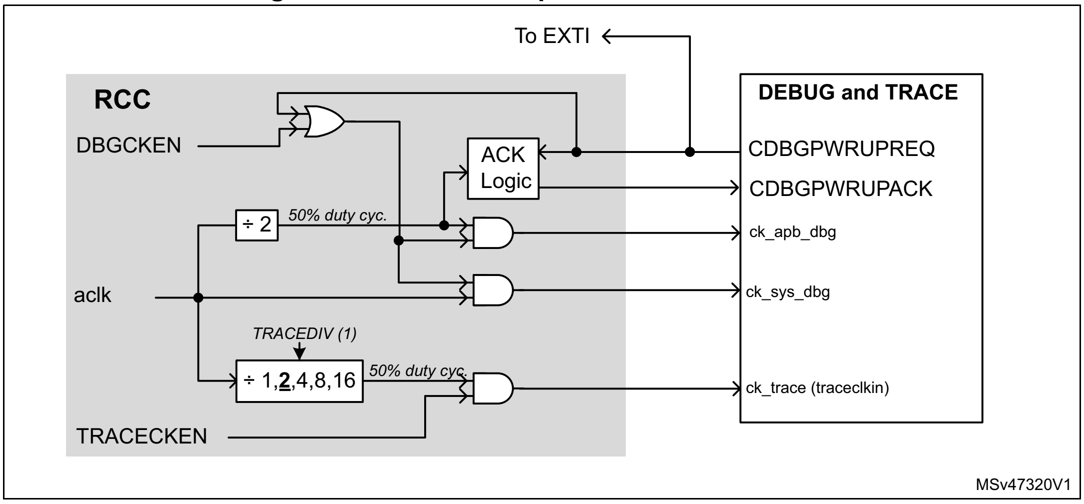
Configuration #8

- The APB clock of DDRC is enabled when the MPU is in CRun, and disabled in CSleep.
- The APB clock of DDRPHYC are enabled when the MPU is in CRun or CSleep.
- The DDRC and the DDRPHYC kernel clocks (**ddrc_ker_ckg** and **dphy_ker_ck**) are gated using to the DDRC-Low-power handshake, according to the MPU mode, and DDRC requests.

Clock distribution for DBG and trace

The clock generation for the trace and debug is controlled by the RCC via the *RCC debug configuration register (RCC_DBGCFGR)*.

Figure 94. Clock description for DBG and trace



1. X Represents the reset value after a system reset.

Clock distribution for IO compensation

The IO compensation cell is used to calibrate the I/Os slew rate. For that purpose, the RCC is providing the **csi_ck** clock to the IO compensation cell block.

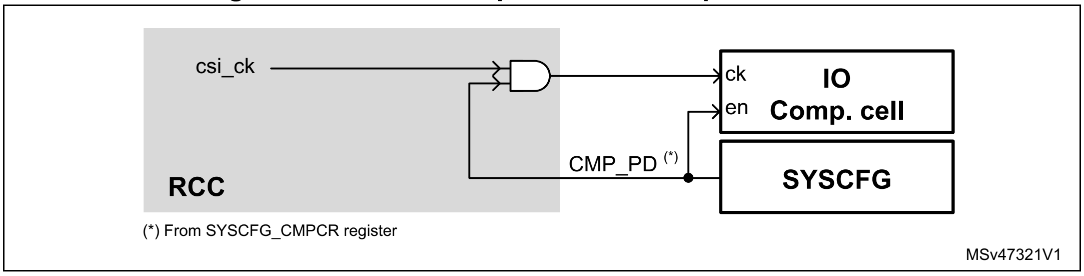
When the compensation mechanism needs to be activated, the clock must be provided to the IO compensation cell block. Meaning that the application has to set the CSION bit to ‘1’.

The **csi_ck** clock is not provided to the IO compensation cell if the CMP_PD bit is set to ‘0’. Refer to *Section 14: System configuration controller (SYSCFG)* block description for further information.

Note that the **csi_ck** clock is automatically gated when the system goes to (LP-)Stop, and if the CSIKERON bit is set to ‘0’. Setting the CSIKERON bit to ‘1’ allows to maintain the **csi_ck** clock enabled even in system (LP-)Stop.

In system Standby the **csi_ck** clock is disabled.

Figure 95. Clock description for IO compensation cell



10.4.9 General clock concept overview

The RCC handles the distribution of the CPUs, bus interface and peripheral clocks for the system, according to the operating mode of each processor. Refer to *Section 9.5.1: PWR operating modes* for details on operating modes.

Refer to *Section 10.4.1: Clock naming convention*, for details on clock definitions.

For most of the peripherals, it is possible for the application to control the activation/deactivation of their kernel and bus interface clocks. Prior to use a peripheral, the MPU or MCU has to enable it (by setting its PERxEN to '1'), and defines if this peripheral remains active in CSleep mode (by setting its PERxLPEN to '1'). This operation is called 'allocate' a peripheral to a processor. Refer to *Section 10.4.10: Peripheral allocation* for more details.

The peripheral allocation is used by the RCC to control automatically the clock gating according to the CPUs modes.

For example if the MPU enables a peripheral and decides that the kernel clock of this peripheral shall be gated when it goes to CSleep, then the state of the MCU has no effect on the clock gating of this peripheral, but the clock gating will obey to the state of the MPU. This is true if the MCU did not allocate this peripheral as well. Allocating a peripheral guarantees to a processor that this peripheral will remain accessible independently of the mode of the other processor.

Each CPU have dedicated registers in order to allocate individually the peripherals they intends to use in the application. Registers dedicated to the MPU are named RCC_MP_xxxx, the registers dedicated to the MCU are named RCC_MC_xxxx.

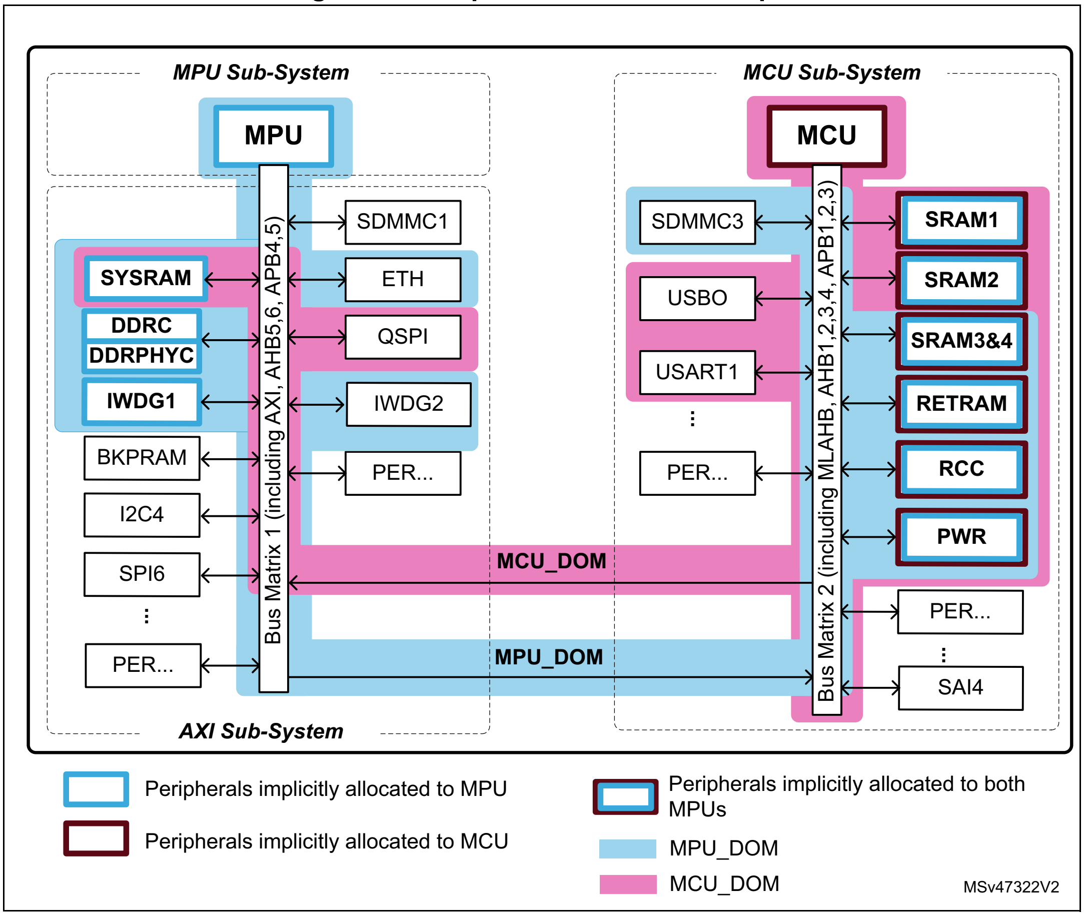
Figure 96 gives an example of peripheral allocation:

- The MPU enabled the ETH, SDMMC3, SRAM3 and RETRAM. Note that SYSRAM, DDRC, DDRPHYC and IWDG1 are implicitly allocated to MPU. The group composed by the MPU, bus matrix 1, bus matrix 2, and peripherals allocated via RCC_MP_xxxx registers are forming the MPU peripheral domain (MPU_DOM).
- The MCU enabled SYSRAM, QUADSPI, USBO and USART1. The group composed by the MCU, bus matrix 1, bus matrix 2 and peripherals allocated via RCC_MC_xxxx registers are forming the MCU peripheral domain (MCU_DOM).

Note: PWR and RCC are common resources and implicitly allocated to both MPU and MCU.

Note: As shown in Figure 96, some memories are implicitly allocated to a processor, refer to Section : Handling of memory devices for more details.

Figure 96. Peripheral allocation example



Note: MPU_DOM and MCU_DOM can be spread over the 2 domains.

When the MPU goes to CStop, then the RCC will disable automatically the bus interface and kernel clocks of all the peripherals of the MPU_DOM as well as the MPU clock. If MCU is still in CRun, then the bus matrix 1 is still clocked as MCU_DOM contains peripherals (SYSRAM, QUADSPI) physically connected to this bus matrix. The PLLs if enabled, are not disabled by the RCC as the MCU is still in running.

Then, if the MCU goes to CStop, the bus matrix 1 is no longer clocked. Note that in this scenario, the complete system can go to Stop. If both CPUs allowed the Standby mode, then the system will go to Standby.

Wakeup events will be able to exit the system from Standby or Stop.

Handling of memory devices

Both CPUs are able to access all the memory devices available in the product:

- SYSRAM,
- SRAM1, SRAM2, SRAM3, SRAM4, RETRAM,
- BKPSRAM.

As shown in *Figure 96*, SYSRAM is implicitly allocated to the MPU. So there is no enable bit for the MPU to allocate this memory. But when the MCU wants to access the SYSRAM, the memory must be enabled via *RCC AXI peripheral enable for MCU set register*

(RCC_MC_AXIMENSETR). Enabling the SYSRAM ensures that this memory will still be operating even if MPU goes to CStop.

However, the SRAM1, SRAM2, SRAM3 and SRAM4 are implicitly allocated to both MCU and MPU because the MLAHB bridge will be enabled if one of the processor is in CRun.

The RETRAM has enable bits for both MPU and MCU. By default this memory is allocated (i.e. enabled) to both processors in order to allow accesses when the system exits from Standby.

The BKPSRAM has a dedicated enable bit for each CPU in order to gate the bus interface clock. The CPUs need to enable the BKPSRAM prior to use it.

Refer to *Section 10.4.11: Peripheral clock gating control* and *Section 10.4.12: Processors and interconnect clocks gating* for details on clock enabling.

10.4.10 Peripheral allocation

Each CPU can allocate a peripheral and hence control its kernel and bus interface clock. Each CPU has dedicated registers in order to perform this peripheral allocation. A peripheral is allocated when its PERxEN bit is set to ‘1’, by the MPU and/or MCU.

The MPU can allocate or deallocate a peripheral for itself by setting the dedicated PERxEN bit on registers RCC_MP_xxxxENSETR and RCC_MP_xxxxENCLRR. The MPU can perform the same operations for the MCU via RCC_MC_xxxxENSETR and RCC_MC_xxxxENCLRR registers.

The MCU can also allocate or deallocate a peripheral for itself by setting the dedicated PERxEN bit on RCC_MC_xxxxENSETR and RCC_MC_xxxxENCLRR registers.

However, the MCU cannot allocate or deallocate a peripheral for the MPU: it can read the RCC_MP_xxxxENSETR and RCC_MP_xxxxENCLRR registers, but the RCC prevents the MCU to write them.

A similar mechanism is implemented for the control of the peripheral clocks when the CPUs are in CSleep (PERxLPEN bits).

The peripheral allocation bits (PERxEN bits) are used by the hardware in order to provide the kernel and bus interface clocks to the peripherals, but they are also used to link peripherals to a CPU (CPU sub-system). In this way, the hardware will be able to safely gate the peripheral clocks and bus matrix clocks, according to CPU states. The PWR block will also use this information in order to control properly the domain states.

Clock enabling delays

In order to avoid spurs, the clock gating logic will synchronize the enable command (coming generally from a kernel clock request or PERxEN bits) with the selected clock.

- A maximum delay of 2 periods of the enabled clock may occur between the enable command, and the first rising edge of the clock. The enable command can be the rising edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request asserted by a peripheral.
- A maximum delay of 1.5 periods of the disabled clock may occur between the disable command, and the last falling edge of the clock. The disable command can be the falling edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request released by a peripheral.

Note: Both the kernel clock and the bus interface clock are affected by this re-synchronization delay.

In addition, the clock enabling delay may increase if the application is enabling for the first time, a peripheral which is not located into its sub-system. This is due to the fact that the sub-system where the peripheral is located may not have its sub-system clock ready. The sub-system clock must be switched on before the application can use this peripheral.

For example if the MCU is enabling a peripheral located into the AXI sub-system, while the MPU is in CStop, then the RCC has to provide a clock to the AXI sub-system.

In order to allocate properly a peripheral, the following sequence must be respected:

- Enable the peripheral clocks (i.e. allocate the peripheral) by writing its PERxEN bit to ‘1’ in the RCC_MP_xxxxENSETR or RCC_MC_xxxxENSETR register,
- Read back the RCC_MP_xxxxENSETR or RCC_MC_xxxxENSETR register. This operation can be considered as a dummy read,
- Then the peripheral can be used.

Note: When the bus interface clock is not active, the read or write accesses to the peripheral registers are not supported. A read access will result in reading invalid data. A write access is ignored and does not create any bus errors.

10.4.11 Peripheral clock gating control

As mentioned previously, each peripheral requires a bus interface clock, named **ckg_bus_perx** (for peripheral ‘x’). This clock is an APB, AHB or AXI clock, according to which bus the peripheral is connected.
The clock used as bus interface for peripherals located into MPU domain could be **aclk**, **hclk5, hclk6, pclk4** or **pclk5**, depending on the bus connected to each peripheral. For simplicity, those clocks are named **mpu_bus_ck**. In the same way, the signal named **mcu_bus_ck** represents **hclk[4:1]**, **pclk1**, **pclk2** or **pclk3**, depending on the bus connected to each peripheral of MCU domain.

Some peripherals (SAI, UART...) need also a dedicated clock for their communication interface, this clock is generally asynchronous with respect to the bus interface clock, and is named kernel clock (**kckg_perx**).

Both clocks can be gated according to several conditions detailed hereafter.

As shown in *Figure 97*, the enabling of the kernel and bus interface clocks of each peripheral depends on several input signals:

- MP_PERxEN, MP_PERxLPEN bits
- MC_PERxEN, and MC_PERxLPEN bits
- **mpu_mode** (see *Table 64: MPU mode details*)
- **mcu_mode** (**mcu_sleep** and **mcu_deepsleep** signals)
- The kernel clock request (**req_ker_perx**) of the peripheral itself, when the feature is available.

MP_PERxEN represents the peripheral enable (allocation) bit for the MPU. The MPU can write these bits to ‘1’ via RCC_MP_xxxxENSETR registers.

MC_PERxEN represents the peripheral enable (allocation) bit for MCU. The MCU can write these bits to ‘1’ via RCC_MC_xxxxENSETR registers.

The MPU State is a logic combination of processors states (WFI or WFE) and STPREQ_P[1:0] bits, see *Table 69* for details.

Refer to *Section 10.4.10: Peripheral allocation* for more details.

Figure 97. Peripheral kernel clock enable logic details

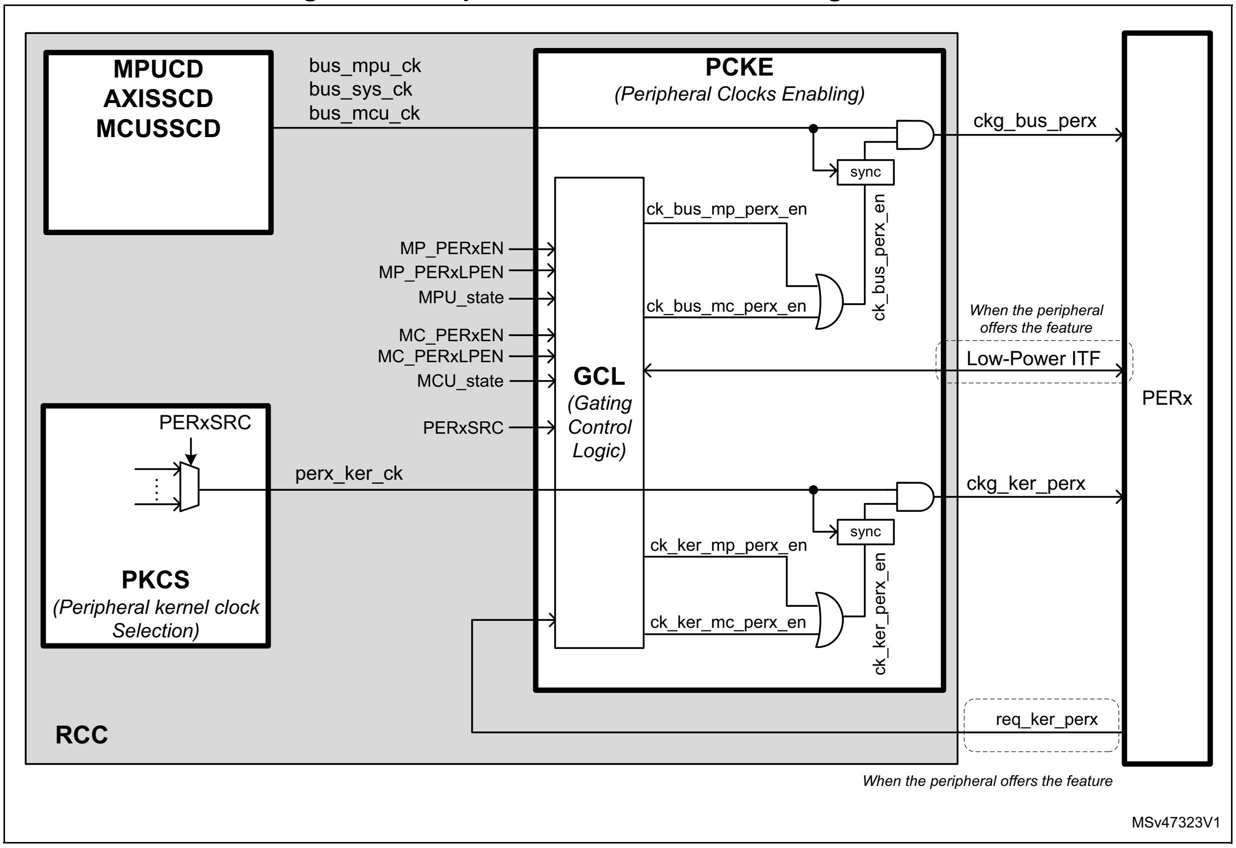


Table 63 gives the details of the enabling logic of most of the peripheral clocks for peripherals allocated by MPU (or MCU).

Table 63. Peripheral clock enable details for MPU (MCU)

MP_PERxEN (MC_PERxEN)	MP_PERxLPEN (MC_PERxLPEN)	PERxSRC	req_ker_perx	MPU State (MCU State)	-	mp_perx_en_ker_ck (mc_perx_en_ker_ck)	(mp_perx_en_bus_ck (mc_perx_en_bus_ck)	Comments
0	X	X	X	X	→	0	0	No clock provided to the peripheral, because PERxEN=0
1	X	X	X	CRun	→	1	1	Kernel and bus interface clocks are provided to the peripheral, because the MPU (MCU) is in CRun, and PERxEN='1'

Table 63. Peripheral clock enable details for MPU (MCU) (continued)

MP_PERxEN (MC_PERxEN)	MP_PERxLPEN (MC_PERxLPEN)	PERxSRC	req_ker_perx	MPU State (MCU State)	-	mp_perx_en_ker_ck (mc_perx_en_ker_ck)	(mp_perx_en_bus_ck (mc_perx_en_bus_ck)	Comments
1	0	X	X	CSleep	→	0	0	No clock provided to the peripheral, because the MPU (MCU) is in CSleep, and PERxLPEN=0
1	1	X	X		→	1	1	Kernel and bus interface clocks are provided to the peripheral, because MPU (MCU) is in CSleep, and PERxLPEN='1'
1	0	X	X	CStop	→	0	0	No clock provided to the peripheral because the PERxLPEN bit is set to '0'.
1	1	no lsi_ck and no lse_ck and no hsi_ker_ck and no csi_ker_ck and no hse_ker_ck	X		→	0	0	No clock provided to the peripheral because MPU (MCU) is in CStop and **lse_ck** or **lsi_ck** or **hsi_ker_ck** or **csi_ker_ck** or **hse_ker_ck** are not selected.
1	1	lsi_ck or lse_ck	X		→	1 [(1)]	0	Kernel clock is provided to the peripheral because PERxEN = PERxLPEN='1' and **lsi_ck** or **lse_ck** are selected. The bus interface clock is no provided as the MPU (MCU) is in CStop. It is up to the application to enable the LSE or LSI if requested.
1	1	hsi_ker_ck or csi_ker_ck or hse_ker_ck	1		→	1	0	Kernel clock is provided to the peripheral because **req_ker_perx** = '1', and PERxEN = PERxLPEN='1' and **hsi_ker_ck** or **csi_ker_ck** or **hse_ker_ck** are selected. The bus interface clock is not provided as the MPU (MCU) is in CStop. If the oscillators are OFF, the RCC will enable them when a kernel clock request occurs.
1	1	hsi_ker_ck or csi_ker_ck or hse_ker_ck	0		→	0	0	No clock provided to the peripheral because MPU (MCU) is in CStop, and no kernel clock request pending

1. Note that for RNG block, the kernel clock is not delivered if the CPU to which it is allocated is in CStop, even if the clock selected is lsi_ck or lse_ck.

Note: The kernel clock request generated by the peripheral, will enable the corresponding oscillator only if the PERxSRC is selecting an oscillator: HSI, CSI or HSE.

As a summary, we can state that the kernel clock is provided to the peripherals in the following conditions:

1. When the CPU to which the peripheral is allocated is in CRun.
2. When the CPU to which the peripheral is allocated is in CSleep and PERxLPEN = ‘1’.
3. When the CPU to which the peripheral is allocated is in CStop, with PERxLPEN = ‘1’, and the peripheral is generating a kernel clock request, and the selected clock is **hsi_ker_ck** or **csi_ker_ck** or **hse_ker_ck**.
4. When the CPU to which the peripheral is allocated is in CStop, with PERxLPEN = ‘1’, and the kernel source clock of the peripheral is **lse_ck** or **lsi_ck**.

The bus interface clock will be provided to the peripherals only when the conditions 1 or 2 are met.

Note: Only some peripherals are able to request the kernel clock: I2C, U(S)ART. This feature gives to the peripheral the capability to wait for an event with an optimal power consumption.

10.4.12 Processors and interconnect clocks gating

CA7-SS (MPU) clock gating

The CA7-SS (MPU) embeds two processors. All the processors and SCU are clocked with the **mpuss_ck** clock provided by the RCC.

When one of the processor goes to Wait For Interrupt (WFI), it is set in low-power state by disabling its clocks, except for the functions used to wake up the processor from WFI.

When one of the processor goes to Wait For Event (WFE), it is set in low-power state by disabling its clocks, except for the functions used to wake up the processor from WFE.

When both processors are in WFI state, the shared L2 memory system logic can also enter in WFI, and the signal **mpu_standbywfi_l2** is asserted.

More precisely, when both processors are in WFI state, then the L2 memory system completes the pending transactions, enters in WFI state, and the signal **mpu_standbywfi_l2** is asserted. If the CStop mode has been requested via STPREQ_Px bits, the RCC disables the **mpuss_ck** clock, and the AXI interface clock (**aclk_mpu**).

Table 64 details the MPU modes according to the processor state (WFI) and RCC register values.

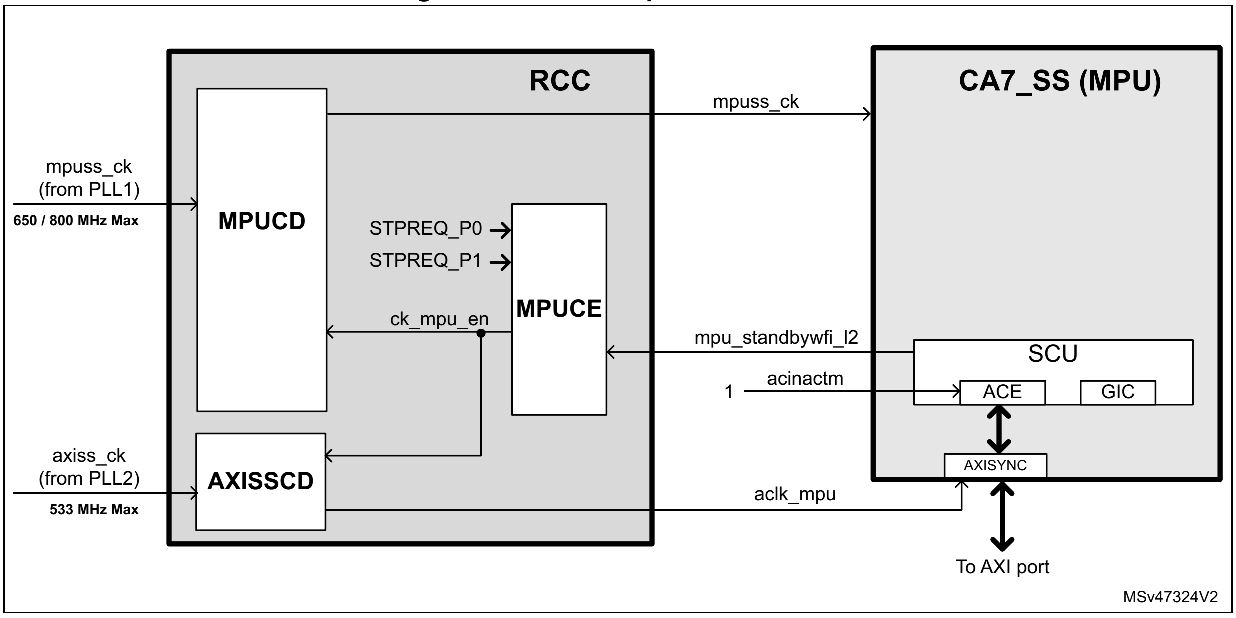
Table 64. MPU mode details (1)

STPREQ_P0 AND STPREQ_P1 (2)	proc_0 in WFI	proc_1 in WFI	-	MPU mode
-	N	-	→	CRun
	-	N	→	
0	Y	Y	→	CSleep
1	Y	Y	→	CStop

1. “-” means don’t care.
2. STPREQ_P0 AND STPREQ_P1 is a logic “AND” between both bits.
The STPREQ_P0 and STPREQ_P1 bits are located into *RCC stop request set register (RCC_MP_SREQSETR)* and *RCC stop request clear register (RCC_MP_SREQCLRR)*.

The clock **mpuss_ck** and **aclk_mpu** are gated by the RCC when the MPU state is CStop.

Figure 98. MPU low-power interface



Bus matrix and bridges clock gating

The bridges and bus matrix clocks are gated according to processors mode, and the peripheral allocated to those processors.

The tables hereafter show in which conditions the RCC is gating the clock bus matrix and bridges.

For information about convention naming, Refer to *Section 10.4.11: Peripheral clock gating control*.

Table 65. Bus clock enable details for AXIM

MPU mode (1)	MCU mode (1)	mpu_peren_axi (2)	mpu_perlpen_axi (2)	mcu_peren_axi (2)	mcu_perlpen_axi (2)	-	AXIM Clock	Description
CRun	Any	X	X	X	X	→	Clocked	The AXIM clock is generated when one of the following condition is true: – The MPU is in CRun – The MPU is in CSleep with at least a peripheral of the AXI-SS allocated by the MPU with the sleep enable bit activated (i.e. MP_ PERxEN = ‘1’ and MP_PERxLPEN = ‘1’) – The MCU is in CRun with at least a peripheral of the AXI-SS allocated (i.e. at least one MC_PERxEN = ‘1’) – The MCU is in CSleep with at least a peripheral of the AXI-SS allocated, with the sleep enable bit activated (i.e. MC_ PERxEN = ‘1’ and MC_PERxLPEN = ‘1’)
CSleep	Any	1	1	X	X	→		
Any	CRun	X	X	1	X	→		
Any	CSleep	X	X	1	1	→		
CSleep	Any	X	0	0	X	→	Gated	In all other cases, the AXIM clock is gated.
	CSleep	X	0	X	0	→		
	CStop	X	0	X	X	→		
CStop / CStandby	Any	X	X	0	X	→		
	CSleep	X	X	X	0	→		
	CStop	X	X	X	X	→		

1. “Any” means any mode. It can be CRun, CSleep, CStop, or CStandby (for MPU only).

2. **mpu_peren_axi** is set to ‘1’ when at least a peripheral connected to the AXI-SS is allocated to the MPU.
mpu_perlpen_axi is set to ‘1’ when at least a peripheral connected to the AXI-SS is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_axi is set to ‘1’ when at least a peripheral connected to the AXI-SS is allocated to the MCU.
mcu_perlpen_axi is set to ‘1’ when at least a peripheral connected to the AXI-SS is allocated to the MCU, with its PERxLPEN bit set to ‘1’.
The AXI-SS represents the AXIM, AHB5, AHB6, APB4, APB5.

Note: When mpu_peren_axi = ‘0’, mpu_perlpen_axi is also equal to ‘0’.

When mpu_peren_axi = mpu_perlpen_axi = ‘1’ it means that at least for one peripheral connected to the AXI-SS, MP_PERxEN = MP_PERxLPEN = ‘1’.

The same applies for MCU.

Table 66. Bus clock enable details for AHB5 and AHB6

MPU mode (1)	MCU mode (1)	mpu_peren_ahb5 (mpu_peren_ahb6) (2)	mpu_perlpen_ahb5 (mpu_perlpen_ahb6) (2)	mcu_peren_ahb5 (mcu_peren_ahb6) (2)	mcu_perlpen_ahb5 (mcu_perlpen_ahb6) (2)	-	AHB5 (AHB6) Clock	Description
CRun	Any	1	X	X	X	→	Clocked	The AHB5 (AHB6) clock is generated when one of the following condition is true: – The MPU is in CRun with at least a peripheral on AHB5 (AHB6) allocated (i.e. at least one MC_PERxEN = ‘1’) – The MPU is in CSleep with at least a peripheral on AHB5 (AHB6) allocated, with the sleep enable bit activated (i.e. MP_ PERxEN = ‘1’ and MP_PERxLPEN = ‘1’) – The MCU is in CRun with at least a peripheral on AHB5 (AHB6) allocated (i.e. at least one MC_PERxEN = ‘1’) – The MCU is in CSleep with at least a peripheral on AHB5 (AHB6) allocated, with the sleep enable bit activated (i.e. MC_ PERxEN = ‘1’ and MC_PERxLPEN = ‘1’)
CSleep	Any	1	1	X	X	→		
Any	CRun	X	X	1	X	→		
Any	CSleep	X	X	1	1	→		
Any	Any	0	X	0	X	→	Gated	In all other cases, the AHB5 (AHB6) clock is gated.
	CSleep	0	X	X	0	→		
	CStop	0	X	X	X	→		
CSleep	Any	X	0	0	X	→		
	CSleep	X	0	X	0	→		
	CStop	X	0	X	X	→		
CStop / CStandby	Any	X	X	0	X	→		
	CSleep	X	X	X	0	→		
	CStop	X	X	X	X	→		

1. “Any” means any mode. It can be CRun, CSleep, CStop, or CStandby (for MPU only).
2. **mpu_peren_ahb5** is set to ‘1’ when at least a peripheral connected to the AHB5 is allocated to the MPU.
mpu_perlpen_ahb5 is set to ‘1’ when at least a peripheral connected to the AHB5 is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_ahb5 is set to ‘1’ when at least a peripheral connected to the AHB5 is allocated to the MCU.
mcu_perlpen_ahb5 is set to ‘1’ when at least a peripheral connected to the AHB5 is allocated to the MCU, with its PERxLPEN bit set to ‘1’.

mpu_peren_ahb6 is set to ‘1’ when at least a peripheral connected to the AHB6 or APB4 is allocated to the MPU.
mpu_perlpen_ahb6 is set to ‘1’ when at least a peripheral connected to the AHB6 or APB4 is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_ahb6 is set to ‘1’ when at least a peripheral connected to the AHB6 or APB4 is allocated to the MCU.
mcu_perlpen_ahb6 is set to ‘1’ when at least a peripheral connected to the AHB6 or APB4 is allocated to the MCU, with its PERxLPEN bit set to ‘1’.

Note: The AXIMC peripheral clock and reset are controlled via RCC_MP_AHB5_xxx registers but this peripheral is not connected to the AHB5.

Due to bus topology, peripherals on the APB4 are accessible via AHB6.

Table 67. Bus clock enable details for MLAHB and AHB4

MPU mode (1)	MCU mode (1)	-	MLAHB (AHB4) Clock	Description
CRun	Any	→	Clocked	The MLAHB and AHB4 clocks are generated when one of the following condition is true: – The MPU is in CRun or CSleep – The MCU is in CRun or CSleep ==> When the system is in Run
CSleep	Any	→		
Any	CRun	→		
Any	CSleep	→		
CStop / CStandby	CStop	→	Gated	-

1. "Any" means any mode: it can be CRun, CSleep, CStop, or CStandby (for MPU only).

Table 68. Bus clock enable details for AHB[3:1]

MPU mode (1)	MCU mode (1)	mpu_peren_ahb1 (mpu_peren_ahb[3:2]) (2)	mpu_perlpen_ahb1 (mpu_perlpen_ahb[3:2]) (2)	mcu_peren_ahb1 (mcu_peren_ahb[3:2]) (2)	mcu_perlpen_ahb1 (mcu_perlpen_ahb[3:2]) (2)	-	AHB1 (AHB[3:2]) Clock	Description
CRun	Any	1	X	X	X	→	Clocked	The AHB1 (AHB[3:2]) clock is generated when one of the following condition is true: – The MPU is in CRun with at least a peripheral on AHB1 (AHB[3:2]) allocated. – The MPU is in CSleep with at least a peripheral on AHB1 (AHB[3:2]) allocated, with the sleep enable bit activated. – The MCU is in CRun with at least a peripheral on AHB1 (AHB[3:2]) allocated. – The MCU is in CSleep with at least a peripheral on AHB1 (AHB[3:2]) allocated, with the sleep enable bit activated.
CSleep		1	1	X	X	→		
Any	CRun	X	X	1	X	→		
	CSleep	X	X	1	1	→		

Table 68. Bus clock enable details for AHB[3:1] (continued)

MPU mode (1)	MCU mode (1)	mpu_peren_ahb1 (mpu_peren_ahb[3:2]) (2)	mpu_perlpen_ahb1 (mpu_perlpen_ahb[3:2]) (2)	mcu_peren_ahb1 (mcu_peren_ahb[3:2]) (2)	mcu_perlpen_ahb1 (mcu_perlpen_ahb[3:2]) (2)	-	AHB1 (AHB[3:2]) Clock	Description
Any	Any	0	X	0	X	→	Gated	In all other cases, the AHB1 (AHB[3:2]) clock is gated
	CSleep		X	X	0	→		
	CStop		X	X	X	→		
CSleep	Any	X	0	0	X	→		
	CSleep	X		X	0	→		
	CStop	X		X	X	→		
CStop / CStandby	Any	X	X	0	X	→		
	CSleep	X	X	X	0	→		
	CStop	X	X	X	X	→		

1. “Any” means any mode: it can be CRun, CSleep, CStop, or CStandby (for MPU only).
2. **mpu_peren_ahb2** is set to ‘1’ when at least a peripheral connected to the AHB2, APB1 or APB2 is allocated to the MPU.
mpu_perlpen_ahb2 is set to ‘1’ when at least a peripheral connected to the AHB2, APB1 or APB2 is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_ahb2 is set to ‘1’ when at least a peripheral connected to the AHB2, APB1 or APB2 is allocated to the MCU.
mcu_perlpen_ahb2 is set to ‘1’ when at least a peripheral connected to the AHB2, APB1 or APB2 is allocated to the MCU, with its PERxLPEN bit set to ‘1’.

mpu_peren_ahb3 is set to ‘1’ when at least a peripheral connected to the AHB3 is allocated to the MPU.
mpu_perlpen_ahb3 is set to ‘1’ when at least a peripheral connected to the AHB3 is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_ahb3 is set to ‘1’ when at least a peripheral connected to the AHB3 is allocated to the MCU.
mcu_perlpen_ahb3 is set to ‘1’ when at least a peripheral connected to the AHB3 is allocated to the MCU, with its PERxLPEN bit set to ‘1’.

mpu_peren_ahb1 is set to ‘1’ when at least a peripheral connected to the AHB1 is allocated to the MPU.
mpu_perlpen_ahb1 is set to ‘1’ when at least a peripheral connected to the AHB1 is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_ahb1 is set to ‘1’ when at least a peripheral connected to the AHB1 is allocated to the MCU.
mcu_perlpen_ahb1 is set to ‘1’ when at least a peripheral connected to the AHB1 is allocated to the MCU, with its PERxLPEN bit set to ‘1’.

Table 69. Bus clock enable details for APB[5:1]

MPU mode (1)	MCU mode (1)	mpu_peren_apb1 (mpu_peren_apb[5:2]) (2)	mpu_perlpen_ahb1 (mpu_perlpen_apb[5:2]) (2)	mcu_peren_ahb1 (mcu_peren_apb[5:2]) (2)	mcu_perlpen_ahb1 (mcu_perlpen_apb[5:2]) (2)	-	APB1 (APB[5:2]) Clock	Description
CRun	Any	1	X	X	X	→	Clocked	The APB1 (APB[5:2]) clock is generated when one of the following condition is true: – The MPU is in CRun with at least a peripheral on APB1 (APB[5:2]) allocated (i.e. at least one MC_PERxEN = ‘1’) – The MPU is in CSleep with at least a peripheral on APB1 (APB[5:2]) allocated, with the sleep enable bit activated (i.e. MP_ PERxEN = ‘1’ and MP_PERxLPEN = ‘1’) – The MCU is in CRun with at least a peripheral on APB1 (APB[5:2]) allocated (i.e. at least one MC_PERxEN = ‘1’) – The MCU is in CSleep with at least a peripheral on APB1 (APB[5:2]) allocated, with the sleep enable bit activated (i.e. MC_ PERxEN = ‘1’ and MC_PERxLPEN = ‘1’)
CSleep	Any	1	1	X	X	→		
Any	CRun	X	X	1	X	→		
Any	CSleep	X	X	1	1	→		
Any	Any	0	X	0	X	→	Gated	In all other cases, the APB1 (APB2,3,4,5) clock is gated
	CSleep	0	X	X	0	→		
	CStop	0	X	X	X	→		
CSleep	Any	X	0	0	X	→		
	CSleep	X	0	X	0	→		
	CStop	X	0	X	X	→		
CStop / CStandby	Any	X	X	0	X	→		
	CSleep	X	X	X	0	→		
	CStop	X	X	X	X	→		

1. “Any” means any mode. It can be CRun, CSleep, CStop, or CStandby (for MPU only).
2. **mpu_peren_apb1** is set to ‘1’ when at least a peripheral connected to the APB1 is allocated to the MPU.
mpu_perlpen_apb1 is set to ‘1’ when at least a peripheral connected to the APB1 is allocated to the MPU, with its PERxLPEN bit set to ‘1’.
mcu_peren_apb1 is set to ‘1’ when at least a peripheral connected to the APB1 is allocated to the MCU.
mcu_perlpen_apb1 is set to ‘1’ when at least a peripheral connected to the APB1 is allocated to the MCU, with its PERxLPEN bit set to ‘1’.

Similar properties apply for mpu_peren_apb[5:2], mpu_perlpen_apb[5:2], mcu_peren_apb[5:2] and mcu_perlpen_apb[5:2].

Remarks

The MLAHB and AHB4 are clocked as soon as one of the CPU is no longer in CStop. This is due to the fact that RCC and PWR blocks are implicitly allocated to both MPU and MCU.

The AXIM can remain clock gated when the MPU is in CStop, and the MCU has no peripheral allocated into a place needing the activated of the AXIM.

10.4.13 Low-power emulation mode

In order to ease the debugging of the circuit, the RCC is able to handle an emulation mode for the (LP-)Stop and Standby modes. Refer to *Section : Low power mode emulation in debug* of DBG section to get additional information.

Note: In debug emulation mode, the clock of the WWDG1 is frozen.

Sleep emulation mode

The Sleep emulation mode is controlled by the DBGSLEEP bit of the DBGMCU_CR register.

When a processor goes to CSleep with DBGSLEEP = ‘1’, then the processor clock, the clocks of all allocated peripherals, debug parts, and interconnect are maintained activated.

Stop emulation mode

The Stop emulation mode is controlled by the DBGSTOP bit of DBGMCU_CR register.

When a processor goes to CStop or when the system goes to Stop with DBGSTOP = ‘1’, then:

- The processor clock, clocks of all allocated peripherals, debug parts, and interconnect are maintained activated,
- the PWR will not signal low-power mode on the PWR_ON and PWR_LP signals. This will keep the V_{DDCORE} domain to be supplied from the regulator in main mode,
- When a wakeup event occurs, then the PWR and EXTI blocks behave normally.
- If it is an exit from system Stop, then the RCC will bypass (or execute very fast) the clock restore sequence as clocks are still there.

When the MPU goes to CStandby, with DBGSTOP = ‘1’, then:

- The MPU clock, clocks of all allocated peripherals, debug parts, and interconnect are maintained activated,
- The interrupt lines connected to the GIC are immediately gated in order to prevent the MPU to execute an exception before it is reset.
- The PWR will not signal the low-power mode on the PWR_ON and PWR_LP signals. This will keep the V_{DDCORE} domain supplied.
- When a wakeup event occurs, then the PWR and EXTI blocks behave normally, and the RCC generates a MPU reset.
- The interrupt lines connected to the GIC are no longer gated.

Standby emulation mode

The Standby emulation mode is controlled by the DBGSTBY bit of the DBGMCU_CR register.

When the system goes to Standby with DBGSTBY = ‘1’, then:

- The processors clocks, clocks of all allocated peripherals, debug parts, and interconnect are maintained activated,
- The interrupt lines connected to the GIC and NVIC are immediately gated in order to prevent the MPU and MCU to execute an exception before it is reset,
- The PWR will keep the PWR_ON active, allowing the V_{DDCORE} domain to remain supplied,
- When a wakeup event occurs, then the PWR and EXTI blocks behave normally, the RCC generates a reset of the complete V_{DDCORE} (**vcore_rst**), but will not reset the debug part (**dbg_rstn** not activated),
- The interrupt lines are no longer gated.

10.4.14 RCC TrustZone function

The RCC TrustZone function is controlled via two bits: TZEN and MCKPROT located into the *RCC TrustZone control register (RCC_TZCR)*.

The bit TZEN allows the RCC and the PWR to be switched in secure mode. The RCC_TZCR register is only accessible by a secure access.
When TZEN = ‘1’, some RCC and PWR registers can only be modified using secure accesses. The reading of registers is always allowed.

The MCKPROT bit allows in addition to control the secure mode for the registers controlling the generation of the **mcuss_ck** clock.

The RCC is able to protect the secure parts from being modified by non-secure accesses. At RCC level the TrustZone security consists on preventing a non-secure device from:

- Changing the settings of the HSE, HSI, CSI, LSE and LSI,
- Changing the settings of the clocks used for the MPU,
- Changing the settings of the clocks for the AXIM, AHB5 and APB5,
- Disabling the peripheral clocks of secure peripherals,
- Reseting the secure peripherals,
- Changing the kernel clock of secure peripherals.

Refer to *Table 74: Register access type versus protection and security* to get the list of impacted registers.

A non-secure write access into a register with the security is enabled will generate a hard fault. Read accesses are always allowed.

*Note: The value of TZEN bit is provided to the PWR block via the signal **rcc_tzen**.*

10.5 RCC interrupts

The RCC provides 2 independent interrupt interfaces:

- An interrupt interface dedicated to the MPU, and
- An interrupt interface dedicated to the MCU.

The interrupt interface dedicated to the MPU, can be switched into secure mode via the TZEN bit. The interrupt interface dedicated to the MCU is always in non-secure mode.

Each interrupt interface provides 2 interrupt lines:

- A general interrupt line (**rcc_mpu_it** or **rcc_mcu_it**), providing events when the PLLs are ready, or when the oscillators are ready.
- An interrupt line dedicated to the wakeup of the processors when they are in CStop (**rcc_mpu_wkup_it** or **rcc_mcu_wkup_it**).

All interrupt lines are active HIGH, and level triggered.

The enabling of the **rcc_mpu_it**, and **rcc_mpu_wkup_it** interrupts are controlled via *RCC clock source interrupt enable register (RCC_MP_CIER)*.

The enabling of the **rcc_mcu_it**, and **rcc_mcu_wkup_it** interrupts are controlled via *RCC clock source interrupt enable register (RCC_MC_CIER)*.

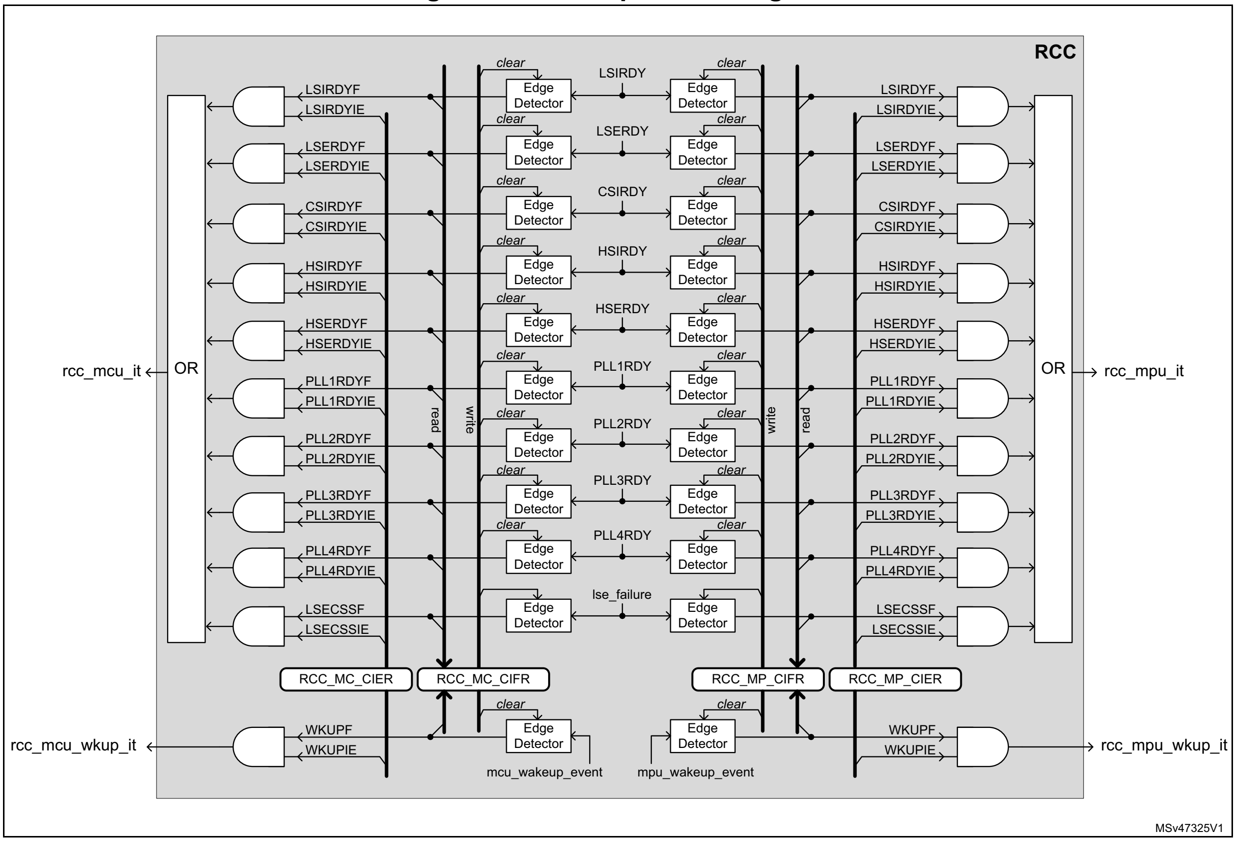
The **rcc_mpu_it** (or **rcc_mcu_it**) line is activated when an event flag is set to ‘1’, and its corresponding enable bit is set to ‘1’ as well. In order to release the **rcc_mpu_it** (or **rcc_mcu_it**) line, all the event flags which have been enabled must be cleared.

The **rcc_mpu_wkup_it** (or **rcc_mcu_wkup_it**) line is activated when the RCC has to exit the MPU (or MCU) from CStop, and if the WKUPIE bit is set to ‘1’ as well. In order to release the **rcc_mpu_wkup_it** (or **rcc_mcu_wkup_it**) line, the MPU (or MCU) shall write the WKUPF bit of the corresponding register to ‘1’.

The interrupt flags can be checked and cleared via *RCC clock source interrupt flag register (RCC_MP_CIFR)* or *RCC clock source interrupt flag register (RCC_MC_CIFR)*.

The event flags goes to ‘1’, on rising edge of the corresponding ready bit (xxxRDY).

Figure 99. Interrupt block diagram



Note: The interrupt flags are relevant even if the corresponding interrupt enable bit is not set.

Table 70 gives a summary of the interrupt sources, and the way to control them.

Table 70. Interrupt sources and control

Interrupt event flag	Description	Interrupt enable	Action to clear interrupt	Interrupt Lines
LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYF to ‘1’	rcc_mcu_it, rcc_mpu_it
LSERDYF	LSE ready	LSERDYIE	Set LSERDYF to ‘1’	
HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYF to ‘1’	
HSERDYF	HSE ready	HSERDYIE	Set HSERDYF to ‘1’	
CSIRDYF	CSI ready	CSIRDYIE	Set CSIRDYF to ‘1’	
PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYF to ‘1’	
PLL2RDYF	PLL2 ready	PLL2RDYIE	Set PLL2RDYF to ‘1’	
PLL3RDYF	PLL3 ready	PLL3RDYIE	Set PLL3RDYF to ‘1’	
PLL4RDYF	PLL4 ready	PLL4RDYIE	Set PLL4RDYF to ‘1’	
LSECSSF	LSE Clock security system failure	LSECSSIE (1)	Set LSECSSF to ‘1’	
WKUPF	Wake up from CStop event	WKUPIE	Set WKUPF to ‘1’	rcc_mpu_wkup_it, rcc_mcu_wkup_it

1. The security system feature must also be enabled (LSECSSON = ‘1’), in order to generate interrupts.

10.6 RCC application information

10.6.1 Handling dynamic clock switching

Care shall be taken when changing on the fly the clock source and clock frequency of processor and bridges:

- The application shall ensure that all the required resources remain clocked during the clock switching in order to ensure that the processor performing the transition remains properly clocked. For example if a processor is doing a clock change which needs to be performed in several steps, the processor, its memory, the bridges and the RCC must remain clocked to allow the software to perform properly the clock transition. If one of these elements is no longer clocked, the transition will not be completed and the system is blocked.
- The application shall also ensure that the frequency of the clocks provided to processors, bridges and peripherals never exceed the maximum allowed value.

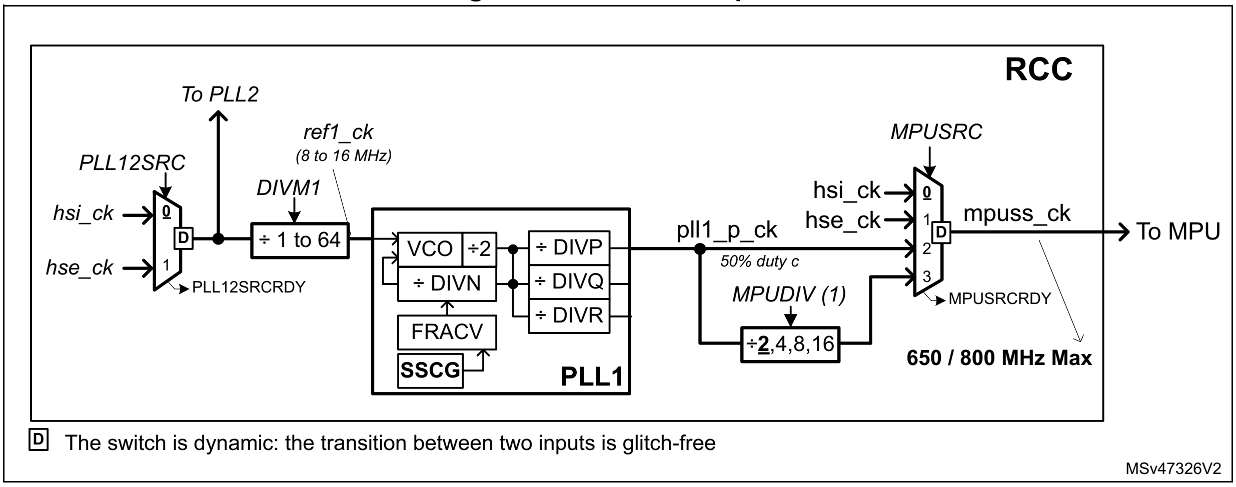
In addition, changing the clock on the interconnect can also affect the performances of some peripheral, more especially if the peripheral is using an APB, AHB or AXI clock as kernel clock.

Changing the MPU clock frequency

The MPU clock frequency can be changed in different ways:

- By changing the MPUSRC value,
- By changing the MPUDIV value,
- By changing the PLL1 settings.

Figure 100. MPU clock path



1. MPUDIV can be changed on-the-fly.
   X Represents the selected MUX input after a system reset.

Changing the MPU clock frequency via MPUSRC

If the MPU clock is changed using MPUSRC, the application has to:

- Ensure that the next clock source is ready before switching the MPUSRC to this input,
- Ensure that the switching of MPUSRC is completed before disabling the previous clock source.

So the procedure is the following:

- Enable the new wanted clock,
- Wait for this clock to be ready,
- Switch the MPUSRC to this new clock,
- Wait until the bit MPUSRCRDY = '1'
- The previous clock source can be disabled.

Changing the MPU clock frequency via MPUDIV

If the MPU clock frequency is changed using MPUDIV, the application has to take care about two things:

- Do not set MPUDIV to ‘000’ if the MPUSRC = ‘3’, otherwise the processor will no longer be clocked,
- Do not exceed the maximum allowed frequency for the **mpuss_ck**.

Changing the MPU clock frequency via PLL1

If the MPU clock frequency is changed via the PLL1, the application has to provide to the MPU a backup clock as long as the transition to the new clock frequency is not complete.

If the MPU is already using the PLL1, the following procedure must be followed:

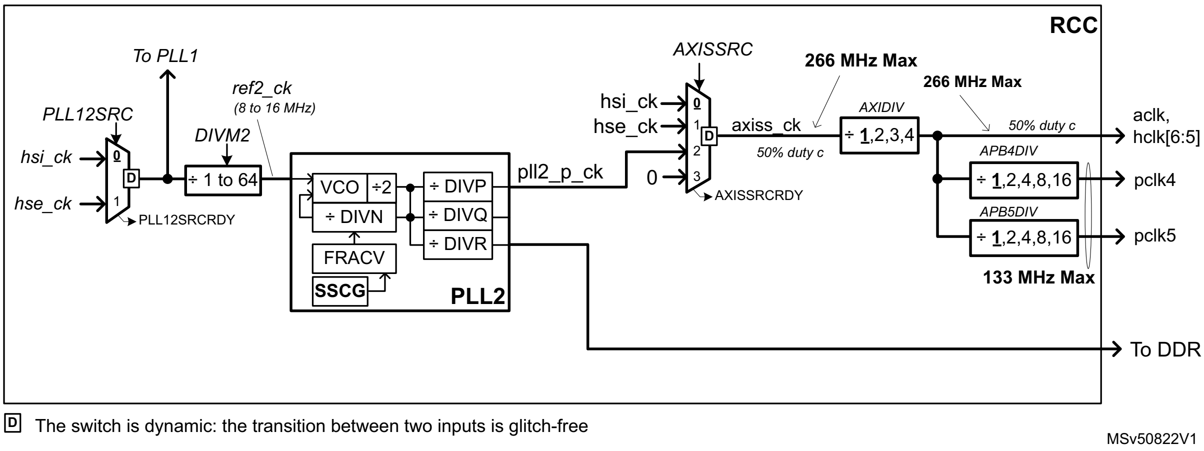
1. Enable a backup clock if not enabled: for example HSI or HSE,
2. Wait for this back-up clock to be ready,
3. Switch the MPUSRC to select this backup clock,
4. Wait for the switching transition to be complete,
5. Change the PLL settings: could be a reprogramming of VCO, which can take few tenth of microseconds due to the PLL lock (see *Section : PLL initialization procedure*), or the reprogramming of the post dividers which is simpler and faster (see *Section : Reprogramming post-dividers*),
6. Wait for the PLL1 to be ready,
7. Switch the MPUSRC to pll1_p_ck,
8. Wait for MPUSRCRDY = '1',
9. Then the back-up clock can be disabled.

Changing the axiss_ck clock source and frequency:

The AXI clock frequency can be changed in different ways:

- By changing the AXISSRC value,
- By changing the AXIDIV value,
- By changing the PLL2 settings.

Figure 101. PLL2 clock path



1. **X** Represents the selected MUX input after a system reset.

Changing axiss_ck frequency via the AXISSRC:

If the **axiss_ck** clock frequency is changed using AXISSRC, the application must do the following:

- Check that the current AXIDIV setting, with the new clock source will not exceed the maximum allowed frequency for the AXIM, AHB[6:5] and APB[5:4]. If it is the case, then:
  - Change the AXIDIV, APB4DIV and APB5DIV in order to ensure that the new clock frequency will not exceed the maximum frequency on AXIM, APB4 and APB5 busses,
  - Wait for AXIDIV, APB4DIV and APB5DIV ready, in order to avoid switching to the new clock source before the new division ratio is effective.
- Generate the new clock source, and wait that this clock source is ready,
- Switch the AXISSRC in order to select the new clock source,
- Wait for AXISSRCRDY = '1' before switching off the previous clock source.

Note: Before switching off the previous clock source be sure that this clock is not used by another sub-system.

Changing axiss_ck frequency via the AXIDIV:

The AXIDIV can be changed on the fly, by simply changing the AXIDIV value.

The application shall ensure that the frequency at the output of AXIDIV will never exceed the maximum allowed frequency for the AXIM, AHB[6:5] and APB[5:4].

The new division value may take some time before being effective, so the application shall always wait for the ready bits of the dividers (AXIDIVRDY, APB[5:4]DIVRDY) to ensure that the new division value is effective.

Changing axiss_ck frequency via the PLL2:

If the **axiss_ck** clock frequency is changed via the PLL2, the application has to generate the **axiss_ck** via a backup clock as long as the transition to the new clock frequency is not complete. If no backup clock is provided, the MPU can no longer run, as the AXI bridge will not be available.

It is possible to change only the DIVP output of the PLL2, by doing the following:

- Enable a back-up clock if not enabled: for example HSI or HSE,
- Wait for this back-up clock to be ready,
- Switch the AXISSRC to this back-up clock,
- Wait for the switching transition to be complete (AXISSRCRDY = ‘1’),
- Check that the current AXIDIV setting, with the new clock source will not exceed the maximum allowed frequency for the AXI. If it is the case, then
  – Change the AXIDIV in order to ensure that the new clock frequency will not exceed the max frequency on AXI bus,
  – Wait for AXIDIV ready, in order to avoid switching to the new clock source before the new division ratio is effective.
- Set the bit DIVPEN of PLL2 to '0',
- Change the value of DIVP,
- Set the bit DIVPEN of PLL2 to '1',
- Switch the AXISSRC to this new clock,
- Wait for AXISSRCRDY = '1',
- Then the back-up clock can be disabled.

Changing the VCO frequency of the PLL2 has more impact, because the PLL2 is also providing the clock to the DDR interface.

In order to change the PLL2 VCO frequency the application has to perform the following steps, assuming that the DDR is used to execute the application code:

- Stop or pause peripherals working with DDR,
- Load the code doing the clock transition into the SYSRAM,
- Give the control to the code located into the SYSRAM.

At this state the DDR shall no longer be used,

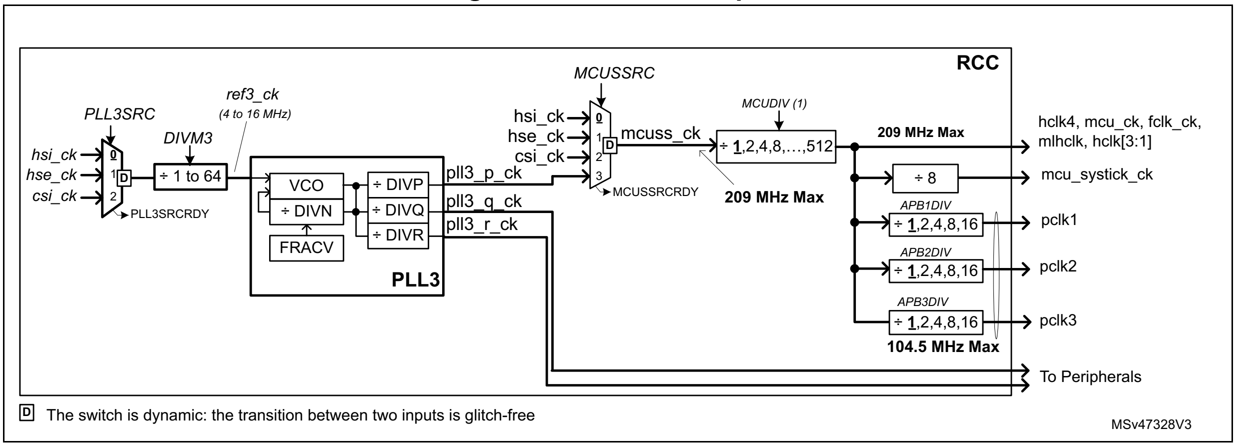
- Set the DDR in self-refresh,
- Enable a back-up clock if not enabled: for example HSI or HSE,
- Wait for this back-up clock to be ready,
- Switch the AXISSRC to this back-up temporary clock (AXISSRC = ‘0’ or ‘1’),
- Wait for the switching transition to be complete,
- Check that the current AXIDIV setting, with the new wanted clock frequency will not exceed the max allowed frequency for the AXI. If it is the case, then:
  – Change the AXIDIV in order to ensure that the new clock frequency will not exceed the max frequency on AXI bus,
  – Wait for AXIDIV ready, in order to avoid switching to the new clock source before the new division ratio is effective.
- Change the PLL settings: see *Section : PLL initialization procedure* for details. This step may take several microseconds due to the PLL lock time,
- Switch the AXISSRC to this new clock (AXISSRC = ‘2’),
- Wait for AXISSRCRDY = '1',
- Then the back-up clock can be disabled,
- The DDR controller shall be re-initialized, because the clock of the DDR interface may have changed due to the change of the frequency of the PLL2 VCO.
- Then the application can switch again on the DDR,
- The peripheral services using DDR can be re-enabled.

Changing the MCU clock source and frequency:

The MCU clock frequency (**mcuss_ck**) can be changed in different ways:

- By changing the MCUSSRC value,
- By changing the MCUDIV value,
- By changing the PLL3 settings.

Figure 102. MCU clock path



1. X Represents the selected MUX input after a system reset.

Changing MCU clock frequency via the MCUSSRC:

If the **mcuss_ck** clock is changed using MCUSSRC, the application must do the following:

- Check that the current MCUDIV setting, with the new clock source will not exceed the max allowed frequency for the MCU domain. If it is the case, then
  - Change MCUDIV, APB1DIV, APB2DIV and APB3DIV in order to ensure that the new clock frequency will not exceed the max frequency on MCU, AHB[4:1], MLAHB and APB[3:1] busses,
  - Wait for MCUDIV, APB1DIV, APB2DIV and APB3DIV ready, in order to avoid switching to the new clock source before the new division ratio is effective.
- Enable the new clock source, and wait that this clock source is ready,
- Switch the MCUSRC to the new clock source,
- Wait for MCUSSRCRDY = '1' before switching off the previous clock source.

Changing MCU clock frequency via the MCUDIV:

The MCUDIV can be changed on the fly, by simply changing the MCUDIV value.

The application shall ensure that the frequency at the output of MCUDIV will never exceed the maximum allowed frequency of the on MCU, AHB[4:1], MLAHB and APB[3:1].

The new division values may take some time before being effective, so the application shall always wait for the ready bits of the dividers (MCUDIVRDY, APB[3:1]DIVRDY) to check when the new division value is effective.

Changing MCU clock frequency via the PLL3:

If the MCU clock frequency is changed via the PLL3, the application has to provide to the MCU a backup clock as long as the transition to the new clock frequency is not complete.

If the MCU is already using the PLL3, the following procedure must be followed:

1. Enable a backup clock if not enabled: for example HSI, CSI or HSE,
2. Wait for this back-up clock to be ready,
3. Switch the MCUSSRC to select this backup clock,
4. Wait for the switching transition to be complete (MCUSSRCRDY = ‘1’)
5. Change the PLL settings: could be a reprogramming of VCO, which can take few tenth of microseconds due to the PLL lock (see *Section : PLL initialization procedure*), or the reprogramming of the post dividers which is simpler and faster (see *Section : Reprogramming post-dividers*).
6. Wait for the PLL3 to be ready,
7. Switch the MCUSSRC to select pll3_p_ck,
8. Wait for the switching transition to be complete (MCUSSRCRDY = ‘1’),
9. Then the back-up clock can be disabled.

10.6.2 PLL programming

PLL initialization procedure

The recommended initialization sequence of the PLLs for the integer and fractional mode is given hereafter:

- The PLL is supposed to be disabled: DIVPEN, DIVQEN, DIVREN, PLLON, and PLLxRDY set to '0', if it is not the case, refer to *PLL disabling procedure*.
- Enable the wanted clock source (HSE, HSI, or CSI) and wait for the ready flag.
- Select the clock source: via RCC_RCK12SELR, RCC_RCK3SELR and RCC_RCK4SELR registers.
- Wait for the clock switch to be ready.
- Initialize the pre-dividers (DIVMx) in order to provide to the PLL, a valid reference clock. DIVMx are located into the RCC_PLLxCFGR1 registers.
- Configure the PLL:
  - In integer or clock spreading mode the application shall ensure that a '0' is loaded into the SDM. This can be done as follow:

    \- Set FRACLE to '0'

    \- Write FRACV to '0'

    \- Set FRACLE to '1'
  - In fractional mode, the application shall load the SDM with the wanted value (FracInitValue) by executing a sequence similar to the integer mode, but with FRACV = FracInitValue.
  - Program the fields IFRGE (for PLL3 and 4), SSCG_CTRL(for PLL 1 and 2), DIVN, DIVP, DIVQ and DIVR.
  - If the clock spreading mode is used (for PLL1 and 2), program also the fields MOD_PER, INC_STEP and SSCG_MODE.
- Enable the PLL:
  - Set the corresponding PLLON bit to '1', and wait for PLLxRDY bit set to '1',
  - If the PLL is in fractional mode, the FRACLE bit must not be set back to '0' as long as PLLxRDY = '0'.
- When the PLLxRDY bit is equal to '1', the application can enable the wanted outputs by setting DIVPEN, DIVQEN, DIVREN to '1'.
- If the application intends to tune the PLL frequency on-the-fly (only possible in fractional mode), then:
  - Set FRACLE to '0',
    When FRACLE = '0', the sigma delta modulator is still operating with the current value,
  - Write the new value into FRACV (FracValue(n)),
  - Set FRACLE to '1', in order to latch the content of FRACV into the SDM.

Note: *When the PLLxRDY goes to '1', it means that the PLLx output frequency is within 2% of its target value.*

Reprogramming post-dividers

When the PLLs are enabled, it is possible to change the values of a post-dividers (DIVP, DIVQ or DIVR) without disabling the corresponding PLL, by performing the following sequence:

- Disable the wanted output by setting the bit DIVPEN, DIVQEN or DIVREN to '0',
- Change the value of the corresponding post-divider (DIVP, DIVQ or DIVR),
- Re-enable this output by setting the bit DIVPEN, DIVQEN or DIVREN to '1'.

Using the SSCG block

If an application needs to generate with the PLL2, a clock signal having the following characteristics

- An output frequency (**Fpll2_r_ck**), as close as possible to 533 MHz, but not more,
- With a modulation depth of 0.5% peak (M_D),
- With a frequency modulation of 25 kHz (F_{MOD}).

The assumption is that the crystal oscillator is 24 MHz (F_{hse_ck}).

DIVM2 is programmed to 2 in order to provide a reference clock of 12 MHz to the PLL2 (F_{ref2_ck}).

From a 12 MHz reference clock, it is possible to generate 528 MHz, using the division by two provided by the PLL2, in order to ensure an optimal duty cycle.

The application has to compute the following parameters:

Computing NDIV by combining the formulas given in *Section : Using the PLLs in integer mode*:

$$DIVN = \frac{F_{pll\_r\_ck}}{F_{ck\_ref}} - 1 = \frac{528}{12} - 1 = 43$$

Computing MOD_PER:

$$MOD\_PER = ROUND\left(\frac{F_{ref\_ck}}{4 \times F_{MOD}}\right) = ROUND\left(\frac{12 \times 10^6}{4 \times 25 \times 10^3}\right) = 120$$

Computing INC_STEP:

$$INC\_STEP = ROUND\left(\frac{(2^{15} - 1) \times M_D \times (DIVN + 1)}{100 \times 5 \times MOD\_PER}\right) = ROUND\left(\frac{32767 \times 0.5 \times (43 + 1)}{100 \times 5 \times 120}\right) = 12$$

Check that MOD_PER x INC_STEP is lower than (2^{15}-1)

In the example, MOD_PER x INC_STEP = 1440 ==> OK

Due to rounding operations, the modulation period and the modulation depth will not completely match the target. In order to choose between center-spread or down-spread modulation we have to check if the frequency max will exceed our requirement of 533 MHz.

The real modulation depth is:

$$M_D\ (\%) = \frac{MOD\_PER \times INC\_STEP \times 100 \times 5}{(2^{15} - 1) \times (DIVN + 1)} = \frac{120 \times 12 \times 100 \times 5}{(2^{15} - 1) \times (43 + 1)} = 0.499\ \%$$

If the center-spread modulation is used, F_{MAX} will be:

$F_{MAX} = F_C$ x (1 + M_D / 100) = 528 x (1 + 0.499 / 100) = 530.6 MHz.

So the center-spread modulation can be used.

PLL disabling procedure

The recommended disabling sequence is given hereafter:

- Disable all the post-dividers: set DIVPEN, DIVQEN and DIVREN bits of PLLx to ‘0’,
- Disable the PLL: set PLLON bit of PLLx to ‘0’,
- Wait for PLLxRDY = ‘0’.

MPU frequency plan examples

The PLL1 is dedicated to the generation of the MPU clock. The application shall provide to the MPU a clock having the following characteristics:

- A duty cycle of 50%,
- A jitter as small as possible, more especially when the MPU frequency is closed to its maximum.

The table hereafter show possible frequency plans for MPU with most usual crystals.

Table 71. Clock settings examples for MPU

HSE (MHz)	PLL1						MPUSRC	mpuss_ck (MHz)	Comments
	DIVM1+1	DIVN+1	DIVP+1	F_{COMP} (MHz)	F_{VCO} (MHz)	F_{fout1_ck} (MHz)			
16	2	100	1	8	1600	800	pll1_p_ck	800	List of all possible frequencies by changing MPUDIV, with MPUSRC = ‘11’: 800, 400, 200, 100, 50 MHz
	2	81		8	1296	648		648	List of all possible frequencies by changing MPUDIV, with MPUSRC = ‘11’: 648, 324, 162, 81, 40.5 MHz
	1	40		16	1280	640		640	List of all possible frequencies by changing MPUDIV, with MPUSRC = ‘11’: 640, 320, 160, 80, 40 MHz
	2	75		8	1200	600		600	List of all possible frequencies by changing MPUDIV, with MPUSRC = ‘11’: 600, 300, 150, 75, 37.5 MHz

Table 71. Clock settings examples for MPU (continued)

HSE (MHz)	PLL1						MPUSRC	mpuss_ck (MHz)	Comments
	DIVM1+1	DIVN+1	DIVP+1	F_{COMP} (MHz)	F_{VCO} (MHz)	F_{fout1_ck} (MHz)			
19.2	2	83	1	9.6	1593.6	796.8	pll1_p_ck	796.8	All other frequencies obtained by changing MPUDIV are allowed as well.
	2	67		9.6	1286.4	643.2		643.2	
		63		9.6	1190.4	595.2		595.2	
24	3	100	1	8	1600	800	pll1_p_ck	800	All other frequencies obtained by changing MPUDIV are allowed as well.
	2	54		12	1296	648		648	
	2	50		12	1200	600		600	
26	2	61	1	13	1586	793	pll1_p_ck	793	All other frequencies obtained by changing MPUDIV are allowed as well.
	2	50		13	1300	650		650	
	2	46		13	1196	598		598	

In order to have a duty cycle as close as possible to 50%, and an optimal jitter the following rules shall be respected:

To improve the jitter:

- F_{COMP} must be as close as possible to the highest accepted input frequency range of the PLL. For PLL1 and PLL2 it is 16 MHz.
- DIVN value must be as small as possible.
- It is better to use the PLL in integer mode.

To avoid duty cycle degradation

- Prefer using DIVP,Q,R in bypass (DIVP,Q,R = ‘0’) more especially for MPU and DDRC.
- Do not use DIVP,Q,R dividers with odd values more especially the value 3.

Note: Care shall be taken when the HSI is used to provide the reference clock. The application has to take into account the accuracy of the HSI to ensure that it will not exceed the maximum allowed frequency.

AXI and DDR frequency plan examples

Table 72. Clock setting examples for AXI, DDR

HSE (MHz)	PLL2								AXIDIV+1	AXI clock (MHz)	DDR clock (MHz)
	DIVM2+1	DIVN+1	DIVP+1	DIVQ+1	DIVR+1	F_{COMP} (MHz)	F_{VCO} (MHz)	F_{fout2_ck} (MHz)			
16	1	33	2	1	1	16	1056	528	1	264	528
	1	25	2	1	1	16	800	400	1	200	400
19.2	2	55	2	1	1	9.6	1056	528	1	264	528

Table 72. Clock setting examples for AXI, DDR (continued)

HSE (MHz)	**PLL2**								**AXIDIV+1**	AXI clock (MHz)	DDR clock (MHz)
	DIVM2+1	**DIVN+1**	**DIVP+1**	**DIVQ+1**	**DIVR+1**	**F_{COMP} (MHz)**	**F_{VCO} (MHz)**	**F_{fout2_ck} (MHz)**			
24	2	44	2	1	1	12	1056	528	1	264	528
	3	50	2	1	1	8	800	400	1	200	400
	2	53	4	2	2	12	1280	640	1	160	320
	2	44	2	1	2	12	1056	528	1	264	264
	2	28	2	1	2	12	672	336	1	168	168
26	2	41	2	1	1	13	1066	533	1	266	533

10.6.3 Configuring the sub-system clocks

This examples shows the register programming sequence in order to configure properly the sub-system clocks (i.e. **mpuss_ck**, **axiss_ck** and **mcuss_ck**).

In this example, it is assumed that:

- The crystal connected to HSE is 24 MHz,
- Both MPU, and MCU are using HSE as clock source,
- The MPU is working at 600 MHz (PLL1), the MCU at 200 MHz (PLL3), and the AXI is 264 MHz (PLL2),
- It is supposed that **mpuss_ck**, **axiss_ck** and **mcuss_ck** are working with HSI (as it is the case after a system reset).

Here is a recommended programming sequence:

- Enabling of the HSE:
  – Check that HSEON and HSERDY are set to ‘0’ (this is the case after a power-on reset),
  – Set HSEON to ‘1’, and wait for HSERDY = ‘1’. When HSERDY = ‘1’, the clock **hse_ck** is ready to use.
- **refx_ck** clock settings:
  – Set PLL12SRC to ‘1’ in order to select the **hse_ck** for the MPU and AXI and wait for PLL12SRCRDY flag equal to ‘1’,
  – Set DIVM1 and DIVM2 to ‘1’ in order to set the frequency of **ref1_ck** and **ref2_ck** clocks at 12 MHz,
  – Set PLL3SRC to ‘1’ in order to select the **hse_ck** for the MCU sub-system, and wait for PLL3SRCRDY flag equal to ‘1’,
  – Set DIVM3 to 2, in order to provide **ref3_ck** clocks at 8 MHz.
- PLL1 settings and enabling:
  – Ensure that the PLL1 is disabled and all PLL outputs as well, if it is not the case, the sequence described in *Section : PLL disabling procedure*, must be executed,
  – Set DIVN to 49d, in order to have a VCO frequency of 1200 MHz, and thus to have **Ffout1_ck** = 600 MHz,
  – Set DIVP to ‘0’,
  – Set the PLL1 into integer mode:
  – - Set FRACLE to ‘0’
  – - Set FRACV to ‘0’,
  – - Then set FRACLE to ‘1’
  – Disable the clock spreading: SSCG_CTRL = ‘0’
  – Set PLLON to ‘1’
  – Wait for PLL1RDY flag set to ‘1’
  – Set DIVPEN to ‘1’.
- PLL2 settings and enabling:
  – Ensure that the PLL2 is disabled and all PLL outputs as well, if it is not the case, the sequence described in *Section : PLL disabling procedure*, must be executed,
  – Set DIVN to 43d, in order to have a VCO frequency of 1056 MHz, and thus to have **Ffout2_ck** = 528 MHz,
  – Set DIVP to ‘1’ (division by 2),
  – Set the PLL2 into integer mode:
  – - Set FRACLE to ‘0’,
  – - Set FRACV to ‘0’,
  – - Then set FRACLE to ‘1’
  – Set INC_STEP = xx, MOD_PER = yy and SSCG_MODE = ‘1’ (down-spread modulation),
  – Set SSCG_CTRL to ‘1’,
  – Set PLLON to ‘1’,
  – Wait for PLL2RDY flag set to ‘1’,
  – Set DIVPEN to ‘1’.

- PLL3 settings and enabling:
  - Ensure that the PLL3 is disabled and all PLL outputs as well, if it is not the case, the sequence described in *Section : PLL disabling procedure*, must be executed,
  - Set DIVN to 49d, in order to have a VCO frequency of 400 MHz,
  - Set DIVP to ‘1’ in order to have a division by 2,
  - Set IFRGE to ‘x1’,
  - Set the PLL3 into integer mode:
  - - Set FRACLE to ‘0’,
  - - Set FRACV to ‘0’,
  - - Then set FRACLE to ‘1’
  - Set PLLON to ‘1’,
  - Wait for PLL3RDY flag set to ‘1’,
  - Set DIVPEN to ‘1’.
- Interconnect clock settings:

  Before increasing the frequency of the **mcuss_ck**, **axiss_ck** and **mpuss_ck**, the application must ensure that the bus matrix clock dividers are properly programmed in order to avoid invalid frequencies.
  - Set MCUDIV to ‘0’ in order to provide a clock of 200 MHz to the MCU,
  - Set AXIDIV to ‘0’ in order to get the AXI frequency of 264 MHz,
  - Set APB4DIV to a value different from 0 to ensure that **Fpclk4** will be lower or equal to 133 MHz,
  - Set APB5DIV to a value bigger than 1 to ensure that **Fpclk5** will be lower or equal to 133 MHz,
  - Set APB1DIV, APB2DIV and APB3DIV to a value different from 0 to ensure that Fpclk[3:1] will be lower or equal to 100 MHz.
- Sub-system clock switching:
  - Set MPUSRC to 2, in order to select the direct DIVP output from PLL1,
  - Set AXISSRC to 2, in order to select the DIVP output from PLL2,
  - Set MCUSSRC to 3, in order to select the DIVP output from PLL3,
  - Wait for MCUSSRCRDY = ‘1’, MPUSRCRDY = ‘1’, and AXISSRCRDY = ‘1’, before disabling the HSI clock, if not used in the system.

10.6.4 The clock calibration using TIMx

It is possible to measure the frequencies of most of clock source generators by means of the input capture of TIMx.

Calibrating the HSI or CSI with the HSE

The RCC provides two dedicated signals hsi_cal_ck and csi_cal_ck in order to perform the calibration of the HSI and CSI oscillators.

The hsi_cal_ck represents the HSI oscillator output, divided by 1024. The csi_cal_ck represents the CSI oscillator output divided by 128.

The signals hsi_cal_ck and csi_cal_ck are connected to TIM12 and TIM15 input capture. For this calibration, it is expected to have the HSE used as the interconnect clock source via

PLL3. The number of interconnect clock periods between consecutive edges of the hsi_cal_ck (or csi_cal_ck) signal provides a measurement of the HSI (or CSI) clock period.

The calibration accuracy depends on the frequency ratio between the TIM15/12 clock, and the hsi_cal_ck (or csi_cal_ck) clock. The greater the ratio is, the more accurate the measurement is.

The frequency of the hsi_cal_ck is typically 62.5 kHz, and 31.25 kHz for csi_cal_ck.

For a timer clock of 50 MHz, the calibration accuracy will be about 0.13% for HSI and 0.065% for CSI.

The HSI and CSI oscillators have dedicated, user-accessible calibration bits for this purpose, see *RCC HSI configuration register (RCC_HSICFGR)* and *RCC CSI configuration register (RCC_CSICFGR)*. When the HSI or CSI are used via the PLLx, it is also possible to fine-tune the interconnect clock using the fractional divider of the PLLs.

Calibrating the HSI or CSI with the LSE

The primary purpose of having the LSE connected to a TIMx input capture is to be able to precisely measure the HSI or CSI. This requires to have the HSI or CSI used as the interconnect clock source either directly or via PLL3. The number of interconnect clock counts between consecutive edges of the LSE signal provides a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) the user can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio): the precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

The HSI and CSI oscillators have dedicated, user-accessible calibration bits for this purpose, see *RCC HSI configuration register (RCC_HSICFGR)* and *RCC CSI configuration register (RCC_CSICFGR)*. When the HSI or CSI are used via the PLLx, it is also possible to fine-tune the interconnect clock using the fractional divider of the PLLs.

Calibrating the LSI with the HSI

It is also possible to measure the LSI frequency: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process deviation: by measuring it versus the HSI clock source, it is possible to determine its frequency with the precision of the HSI. The measured value can be used to have more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy.

10.6.5 Clock restore sequence examples

The following section shows some simplified timing examples of the execution of the clock restore sequence.

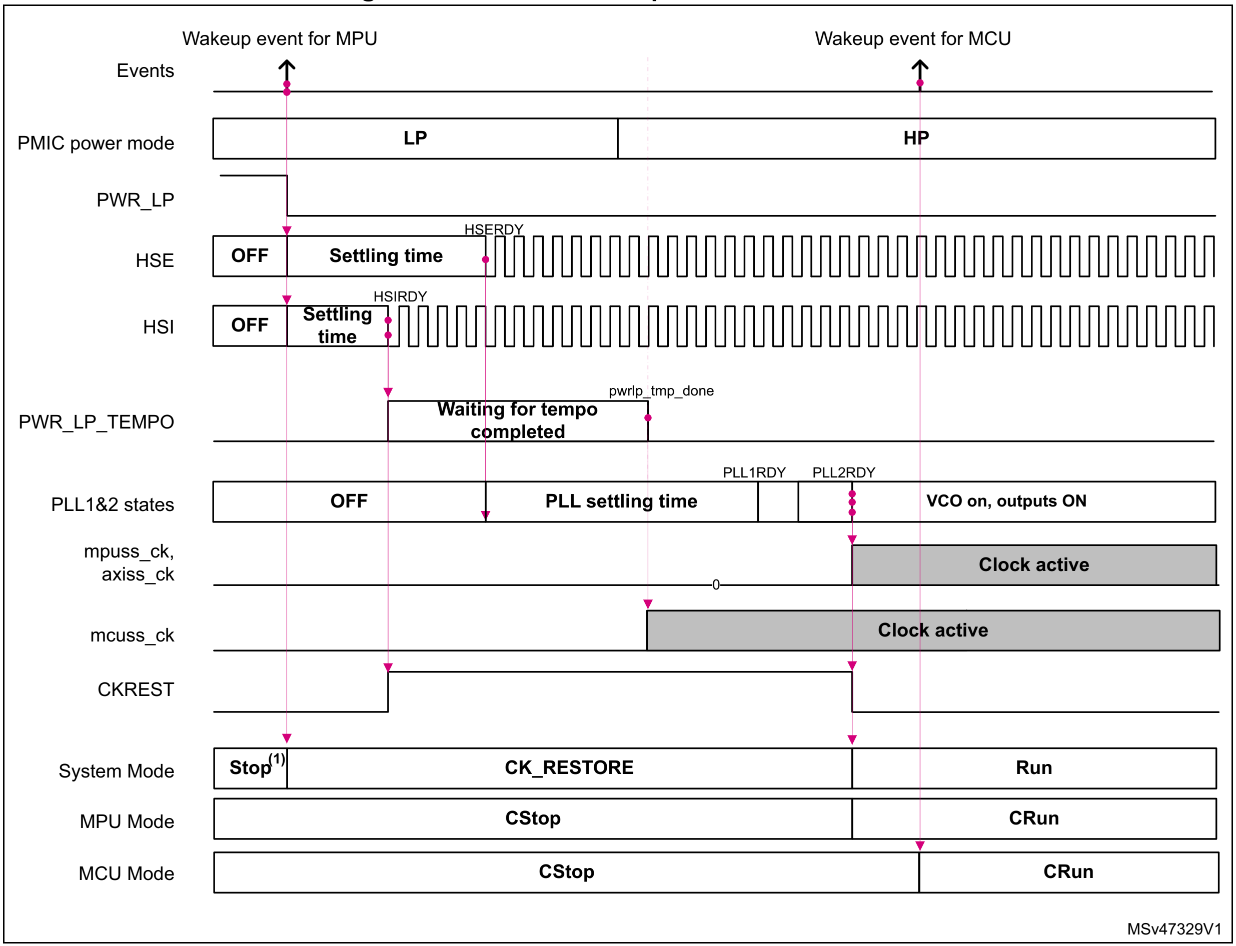
Exit from Stop due to a MPU event

In this example, the system exits from one of the Stop modes due to an event for the MPU, later on, an event for the MCU occurs.

Figure 103 is based on the following configuration:

- The **mpuss_ck** is provided by PLL1 via HSE,
- The **axiss_ck** is provided by PLL2 via HSE,
- The **mcuss_ck** is provided by HSI,
- The bit MCTMPSKP = ‘0’: need to wait for PMIC transition from LP to HP.

Figure 103. Exit from Stop due to MPU event



1. Can be Stop, LP-Stop or LPLV-Stop.

Refer to *Leaving Stop, LP-Stop or LPLV-Stop mode* for additional informations.

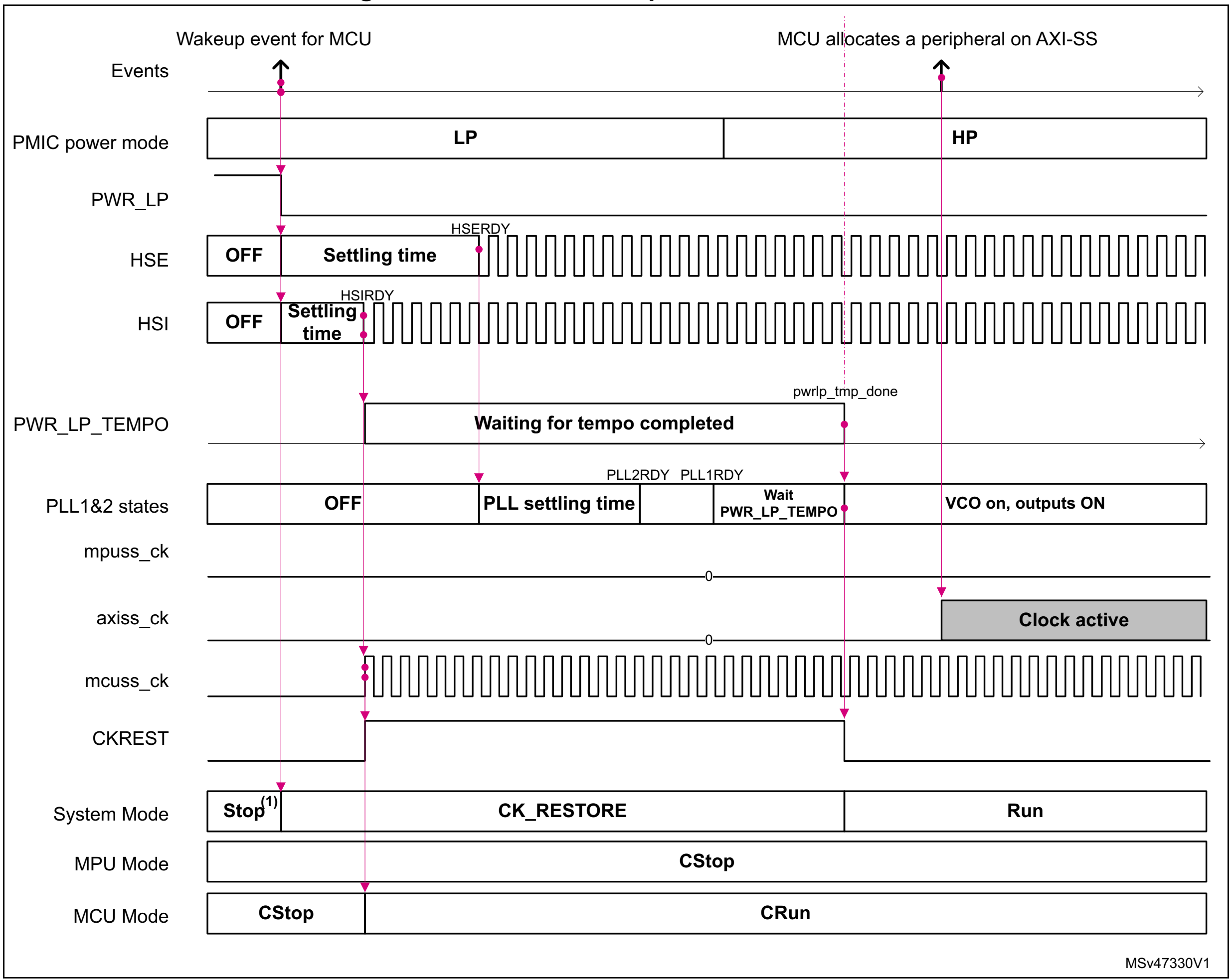
Exit from Stop due to a MCU event, example 1

In this example, the system exits from one of the Stop modes due to an event for the MCU, later the MCU allocates a peripheral on the AXI-SS.

Figure 104 is based on the following configuration:

- The **mpuss_ck** is provided by PLL1 via HSE,
- The **axiss_ck** is provided by PLL2 via HSE,
- The **mcuss_ck** is provided by HSI,
- The MCU does not have peripherals allocated into the AXI-SS,
- The bit MCTMPSKP = ‘1’: no need to wait for PMIC transition from LP to HP.

Figure 104. Exit from Stop due to MCU event



1. Can be Stop, LP-Stop or LPLV-Stop.

Refer to *Leaving Stop, LP-Stop or LPLV-Stop mode* for additional informations.

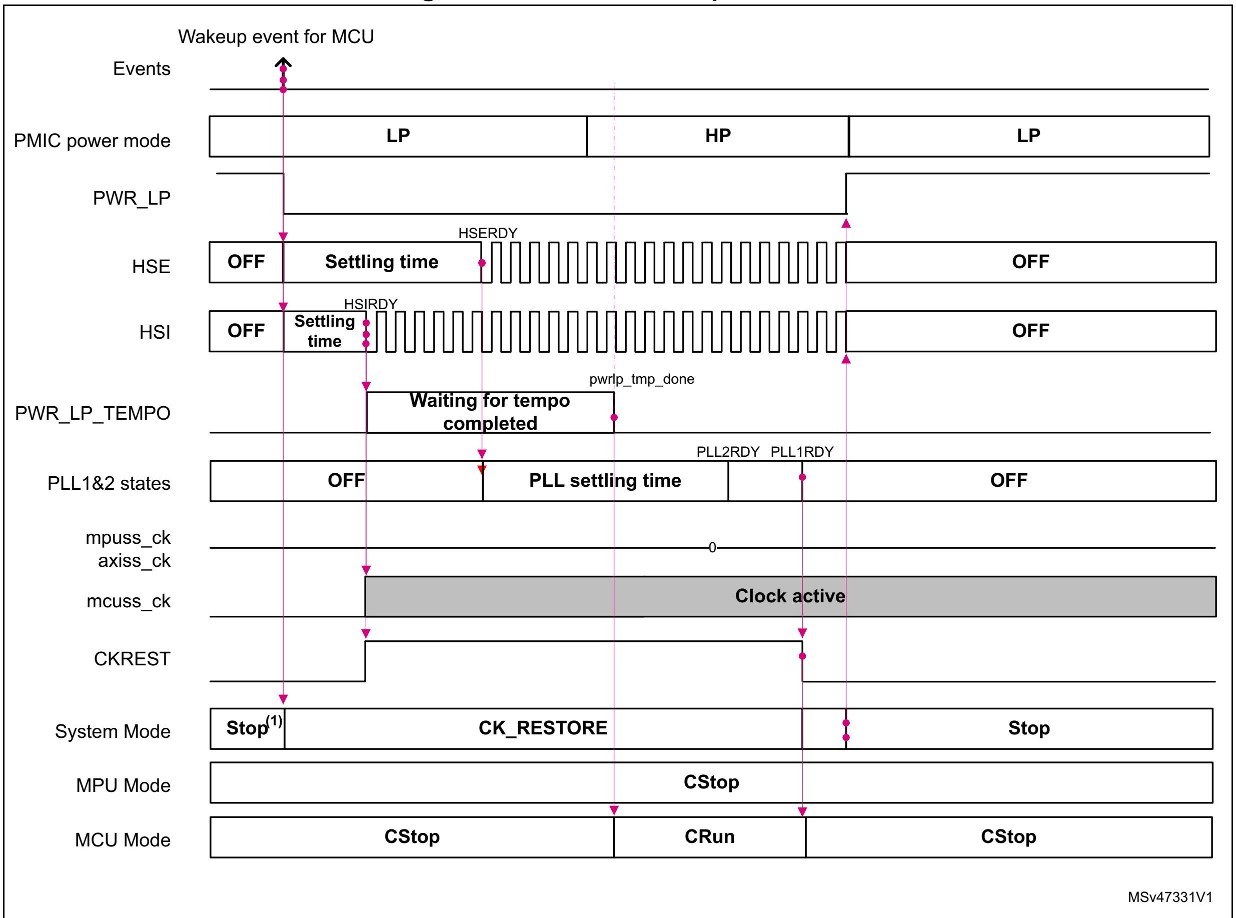
Exit from Stop due to a MCU event, example 2

In this example, the system exits from one of the Stop modes due to an event for the MCU, the MCU executes a task and goes back to CStop. Note that the MCU goes to CStop only when the bit CKREST is set to '0'.

Figure 105 is based on the following configuration:

- The **mpuss_ck** is provided by PLL1 via HSE,
- The **axiss_ck** is provided by PLL2 via HSE,
- The **mcuss_ck** is provided by HSI,
- The MCU does not have peripherals allocated into the AXI-SS,
- The bit MCTMPSKP = '0': need to wait for PMIC transition from LP to HP.

Figure 105. Exit from Stop for MCU



1. Can be Stop, LP-Stop or LPLV-Stop.

Refer to *Leaving Stop, LP-Stop or LPLV-Stop mode* for additional informations.

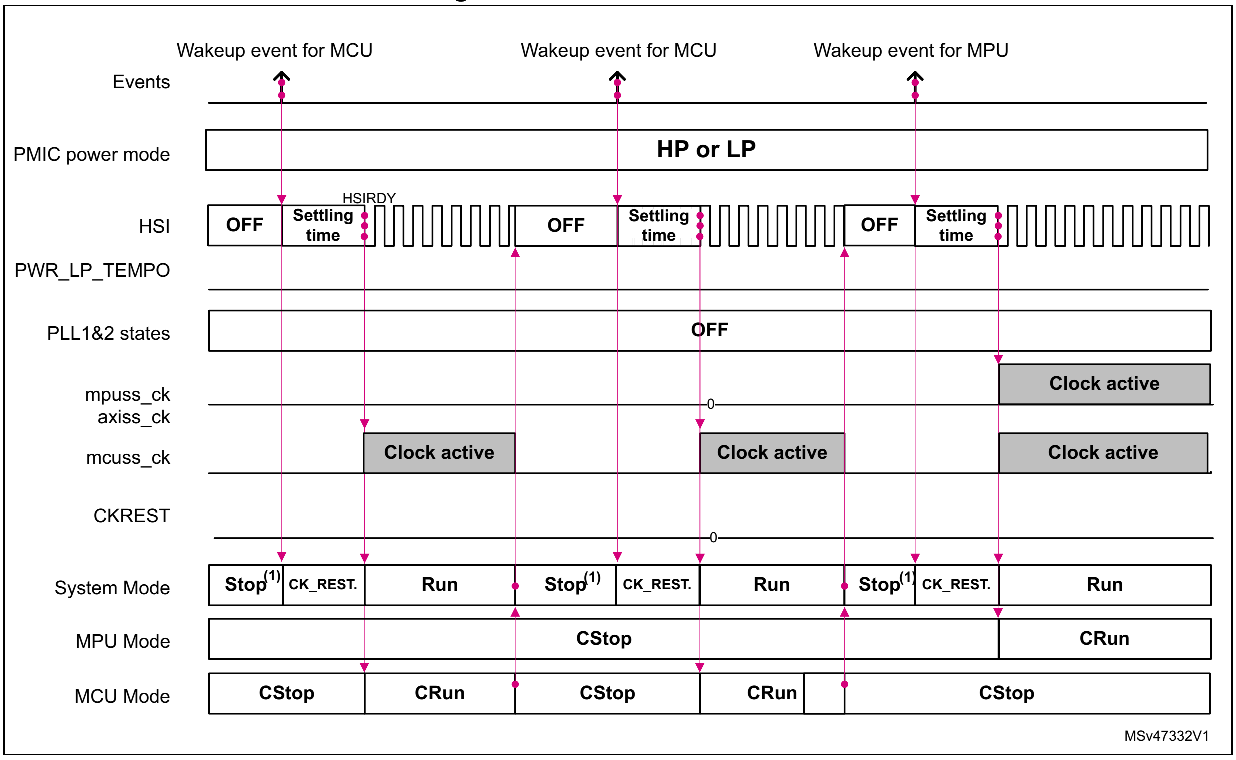
Fast clock restore

In this example, the system exits from one of the Stop modes due to an event for the MCU or MPU. In order to speed-up the occurrence of the CStop-CRun-CStop sequence, the system switch back to HSI as clock source for both MCU, MPU and AXI. In that way, the clock restore sequence is reduced to the activation of the HSI clock.

Figure 106 is based on the following configuration:

- The **mpuss_ck** is provided by HSI before going to CStop
- The **axiss_ck** is provided by HSI before going to CStop
- The **mcuss_ck** is provided by HSI
- The MCU does not have peripherals allocated into the AXI-SS.
- The field PWRLP_DLY is set to ‘0’ in order to bypass the low-power switch tempo.

Figure 106. Fast clock restore



1. Can be Stop, LP-Stop or LPLV-Stop.

Refer to *Leaving Stop, LP-Stop or LPLV-Stop mode* for additional informations.

10.7 RCC registers

The table hereafter gives a summary of the access property of each register. Some registers can be accessed by every bus master, some other can only be accessed by the MPU.

Some registers can only be accessed in secure mode, when the RCC is switched in secure mode.

Table 73 shows the register accesses ("Access Results") according to the possible register attributes. The table shall be read as follow: for a register which can be switched in secure mode ("S"), read operation are always valid, write operations in secure mode are also always valid, write operation in non-secure mode are invalid if TZEN = '1' (or when TZEN = MCKPROT = '1' for registers sensitive to those two bits).

Table 73. Access results according to register security attributes

Secure attribute (1)	TZEN or (TZEN and MCKPROT) (2)	Register access		Access results	Notifications
		Secure	Direction		
AS	x	x	R	Register is read	-
	x	Secure	W	Register is written	-
	x	Non-secure	W	Register is not written	Hard fault
S	x	x	R	Register is read	-
	x	Secure	W	Register is written	-
	0	Non-secure	W	Register is written	-
	1	Non-secure	W	Register is not written	Hard fault
NS	x	x	x	Register is read or written	-

1. "AS" means Alway Secure.
"S" means securable (depends on TZEN and MCKPROT).
"NS" means Non-Secure.

2. For registers controlling the generation of the **mcuss_ck** clock, the security is controlled by an AND between TZEN and MCKPROT.

In addition, some registers can only be modified by the MPU ("MPU Only" in *Table 74*). When the MCU writes into a register having the attribute "MPU Only", the write operation is ignored and a hard fault is triggered. Read accesses to registers having the attribute "MPU Only" are always allowed.

Table 74 gives the attributes for all RCC registers.

Table 74. Register access type versus protection and security

Register name	Access types (1)	Access limitations (2)	Secure attribute (3)	Secure control bits
RCC_OCENCLRR	CR	N	S	TZEN
RCC_OCENSETR	SR	N	S	TZEN and MCKPROT
RCC_HSICFGR	RW	N	S	TZEN

Table 74. Register access type versus protection and security (continued)

Register name	Access types (1)	Access limitations (2)	Secure attribute (3)	Secure control bits
RCC_CSICFGR	RW	N	S	TZEN and MCKPROT
RCC_RCK12SELR	RW	N	S	TZEN
RCC_MPCKSELR	RW	N	S	TZEN
RCC_MPCKDIVR	RW	N	S	TZEN
RCC_ASSCKSELR	RW	N	S	TZEN
RCC_AXIDIVR	RW	N	S	TZEN
RCC_APB4DIVR	RW	N	S	TZEN
RCC_APB5DIVR	RW	N	S	TZEN
RCC_MSSCKSELR	RW	N	S	TZEN and MCKPROT
RCC_MCUDIVR	RW	N	S	TZEN and MCKPROT
RCC_RTCDIVR	RW	N	S	TZEN
RCC_BDCR	RW	N	S	TZEN
RCC_RDLSICR	RW	N	S	TZEN
RCC_PWRLPDLYCR	RW	N	S	TZEN
RCC_PLL1CR	RW	N	S	TZEN
RCC_PLL1CFGR1	RW	N	S	TZEN
RCC_PLL1CFGR2	RW	N	S	TZEN
RCC_PLL1CSGR	RW	N	S	TZEN
RCC_PLL1FRACR	RW	N	S	TZEN
RCC_PLL2CR	RW	N	S	TZEN
RCC_PLL2CFGR1	RW	N	S	TZEN
RCC_PLL2CFGR2	RW	N	S	TZEN
RCC_PLL2CSGR	RW	N	S	TZEN
RCC_PLL2FRACR	RW	N	S	TZEN
RCC_RCK3SELR	RW	N	S	TZEN and MCKPROT
RCC_PLL3CR	RW	N	S	TZEN and MCKPROT
RCC_PLL3CFGR1	RW	N	S	TZEN and MCKPROT
RCC_PLL3CFGR2	RW	N	S	TZEN and MCKPROT

Table 74. Register access type versus protection and security (continued)

Register name	Access types (1)	Access limitations (2)	Secure attribute (3)	Secure control bits
RCC_PLL3CSGR	RW	N	S	TZEN and MCKPROT
RCC_PLL3FRACR	RW	N	S	TZEN and MCKPROT
RCC_I2C46CKSELR	RW	N	S	TZEN
RCC_SPI6CKSELR	RW	N	S	TZEN
RCC_UART1CKSELR	RW	N	S	TZEN
RCC_RNG1CKSELR	RW	N	S	TZEN
RCC_STGENCKSELR	RW	N	S	TZEN
RCC_DDRITFCR	RW	N	S	TZEN
RCC_MP_BOOTCR	RW	MPU only	S	TZEN
RCC_MP_SREQSETR, RCC_MP_SREQCLRR	SR/CR	MPU only	S	TZEN
RCC_MP_GCR	RW	MPU only	S	TZEN
RCC_MP_CIER, RCC_MP_CIFR	RW/CR	N	S	TZEN
RCC_MP_IWDGFZSETR, RCC_MP_IWDGFZCLRR	SR/CR	MPU only	S	TZEN
RCC_BR_RSTSCLRR	CR	N	S	TZEN
RCC_MP_RSTSCLRR, RCC_MP_RSTSSETR	SR/CR	N	S	TZEN
RCC_MP_GRSTCSETR	SR	MPU only	S	TZEN
RCC_APB5RSTSETR, RCC_APB5RSTCLRR	SR/CR	N	S	TZEN
RCC_AHB5RSTSETR, RCC_AHB5RSTCLRR	SR/CR	N	S	TZEN
RCC_TZAHB6RSTSETR, RCC_TZAHB6RSTCLRR	SR/CR	N	S	TZEN
RCC_MP_APB5ENSETR, RCC_MP_APB5ENCLRR	SR/CR	N	S	TZEN
RCC_MP_AHB5ENSETR, RCC_MP_AHB5ENCLRR	SR/CR	N	S	TZEN
RCC_MP_TZAHB6ENSETR, RCC_MP_TZAHB6ENCLRR	SR/CR	N	S	TZEN
RCC_MP_APB5LPENSETR, RCC_MP_APB5LPENCLRR	SR/CR	N	S	TZEN
RCC_MP_AHB5LPENSETR, RCC_MP_AHB5LPENCLRR	SR/CR	N	S	TZEN
RCC_MP_TZAHB6LPENSETR, RCC_MP_TZAHB6LPENCLRR	SR/CR	N	S	TZEN
RCC_TZCR	RW	N	AS	-
RCC_OCRDYR	R	N	NS	-
RCC_MCO1CFGR	RW	N	NS	-
RCC_MCO2CFGR	RW	N	NS	-
RCC_APB1DIVR	RW	N	NS	-
RCC_APB2DIVR	RW	N	NS	-
RCC_APB3DIVR	RW	N	NS	-

Table 74. Register access type versus protection and security (continued)

Register name	Access types (1)	Access limitations (2)	Secure attribute (3)	Secure control bits
RCC_RCK4SELR	RW	N	NS	-
RCC_PLL4CR	RW	N	NS	-
RCC_PLL4CFGR1	RW	N	NS	-
RCC_PLL4CFGR2	RW	N	NS	-
RCC_PLL4FRACR	RW	N	NS	-
RCC_PLL4CSGR	RW	N	NS	-
RCC_TIMG1PRER	RW	N	NS	-
RCC_TIMG2PRER	RW	N	NS	-
RCC_I2C12CKSELR	RW	N	NS	-
RCC_I2C35CKSELR	RW	N	NS	-
RCC_SAI1CKSELR	RW	N	NS	-
RCC_SAI2CKSELR	RW	N	NS	-
RCC_SAI3CKSELR	RW	N	NS	-
RCC_SAI4CKSELR	RW	N	NS	-
RCC_SPI2S1CKSELR	RW	N	NS	-
RCC_SPI2S23CKSELR	RW	N	NS	-
RCC_SPI45CKSELR	RW	N	NS	-
RCC_UART6CKSELR	RW	N	NS	-
RCC_UART24CKSELR	RW	N	NS	-
RCC_UART35CKSELR	RW	N	NS	-
RCC_UART78CKSELR	RW	N	NS	-
RCC_SDMMC12CKSELR	RW	N	NS	-
RCC_SDMMC3CKSELR	RW	N	NS	-
RCC_ETHCKSELR	RW	N	NS	-
RCC_USBCKSELR	RW	N	NS	-
RCC_QSPICKSELR	RW	N	NS	-
RCC_FMCCKSELR	RW	N	NS	-
RCC_FDCANCKSELR	RW	N	NS	-
RCC_SPDIFCKSELR	RW	N	NS	-
RCC_CECCKSELR	RW	N	NS	-
RCC_RNG2CKSELR	RW	N	NS	-
RCC_ADCCKSELR	RW	N	NS	-
RCC_LPTIM45CKSELR	RW	N	NS	-
RCC_LPTIM23CKSELR	RW	N	NS	-

Table 74. Register access type versus protection and security (continued)

Register name	Access types (1)	Access limitations (2)	Secure attribute (3)	Secure control bits
RCC_LPTIM1CKSELR	RW	N	NS	-
RCC_CPERCKSELR	RW	N	NS	-
RCC_DBGCFGR	RW	N	NS	-
RCC_MC_RSTSCLRR	CR	N	NS	-
RCC_APB1RSTSETR, RCC_APB1RSTCLRR	SR/CR	N	NS	-
RCC_APB2RSTSETR, RCC_APB2RSTCLRR	SR/CR	N	NS	-
RCC_APB3RSTSETR, RCC_APB3RSTCLRR	SR/CR	N	NS	-
RCC_APB4RSTSETR, RCC_APB4RSTCLRR	SR/CR	N	NS	-
RCC_AHB6RSTSETR, RCC_AHB6RSTCLRR	SR/CR	N	NS	-
RCC_AHB2RSTSETR, RCC_AHB2RSTCLRR	SR/CR	N	NS	-
RCC_AHB3RSTSETR, RCC_AHB3RSTCLRR	SR/CR	N	NS	-
RCC_AHB4RSTSETR, RCC_AHB4RSTCLRR	SR/CR	N	NS	-
RCC_MP_APB1ENSETR, RCC_MP_APB1ENCLRR	SR/CR	N	NS	-
RCC_MP_APB2ENSETR, RCC_MP_APB2ENCLRR	SR/CR	N	NS	-
RCC_MP_APB3ENSETR, RCC_MP_APB3ENCLRR	SR/CR	N	NS	-
RCC_MC_APB4ENSETR, RCC_MC_APB4ENCLRR	SR/CR	N	NS	-
RCC_MC_APB5ENSETR, RCC_MC_APB5ENCLRR	SR/CR	N	NS	-
RCC_MP_AHB6ENSETR, RCC_MP_AHB6ENCLRR	SR/CR	N	NS	-
RCC_MP_AHB2ENSETR, RCC_MP_AHB2ENCLRR	SR/CR	N	NS	-
RCC_MP_AHB3ENSETR, RCC_MP_AHB3ENCLRR	SR/CR	N	NS	-
RCC_MP_AHB4ENSETR, RCC_MP_AHB4ENCLRR	SR/CR	N	NS	-
RCC_MC_AHB5ENSETR, RCC_MC_AHB5ENCLRR	SR/CR	N	NS	-
RCC_MP_MLAHBENSETR, RCC_MP_MLAHBENCLRR	SR/CR	N	NS	-
RCC_MC_APB1ENSETR, RCC_MC_APB1ENCLRR	SR/CR	N	NS	-
RCC_MC_APB2ENSETR, RCC_MC_APB2ENCLRR	SR/CR	N	NS	-
RCC_MC_APB3ENSETR, RCC_MC_APB3ENCLRR	SR/CR	N	NS	-
RCC_MP_APB4ENSETR, RCC_MP_APB4ENCLRR	SR/CR	N	NS	-
RCC_MC_AHB6ENSETR, RCC_MC_AHB6ENCLRR	SR/CR	N	NS	-
RCC_MC_AHB2ENSETR, RCC_MC_AHB2ENCLRR	SR/CR	N	NS	-
RCC_MC_AHB3ENSETR, RCC_MC_AHB3ENCLRR	SR/CR	N	NS	-
RCC_MC_AHB4ENSETR, RCC_MC_AHB4ENCLRR	SR/CR	N	NS	-
RCC_MC_AXIMENSETR, RCC_MC_AXIMENCLRR	SR/CR	N	NS	-
RCC_MC_MLAHBENSETR, RCC_MC_MLAHBENCLRR	SR/CR	N	NS	-
RCC_MP_APB1LPENSETR, RCC_MP_APB1LPENCLRR	SR/CR	N	NS	-

Table 74. Register access type versus protection and security (continued)

Register name	Access types (1)	Access limitations (2)	Secure attribute (3)	Secure control bits
RCC_MP_APB2LPENSETR, RCC_MP_APB2LPENCLRR	SR/CR	N	NS	-
RCC_MP_APB3LPENSETR, RCC_MP_APB3LPENCLRR	SR/CR	N	NS	-
RCC_MC_APB4LPENSETR, RCC_MC_APB4LPENCLRR	SR/CR	N	NS	-
RCC_MC_APB5LPENSETR, RCC_MC_APB5LPENCLRR	SR/CR	N	NS	-
RCC_MP_AHB6LPENSETR, RCC_MP_AHB6LPENCLRR	SR/CR	N	NS	-
RCC_MP_AHB2LPENSETR, RCC_MP_AHB2LPENCLRR	SR/CR	N	NS	-
RCC_MP_AHB3LPENSETR, RCC_MP_AHB3LPENCLRR	SR/CR	N	NS	-
RCC_MP_AHB4LPENSETR, RCC_MP_AHB4LPENCLRR	SR/CR	N	NS	-
RCC_MP_AXIMLPENSETR, RCC_MP_AXIMLPENCLRR	SR/CR	N	NS	-
RCC_MP_MLAHBLPENSETR, RCC_MP_MLAHBLPENCLRR	SR/CR	N	NS	-
RCC_MC_APB1LPENSETR, RCC_MC_APB1LPENCLRR	SR/CR	N	NS	-
RCC_MC_APB2LPENSETR, RCC_MC_APB2LPENCLRR	SR/CR	N	NS	-
RCC_MC_APB3LPENSETR, RCC_MC_APB3LPENCLRR	SR/CR	N	NS	-
RCC_MP_APB4LPENSETR, RCC_MP_APB4LPENCLRR	SR/CR	N	NS	-
RCC_MC_AHB6LPENSETR, RCC_MC_AHB6LPENCLRR	SR/CR	N	NS	-
RCC_MC_AHB2LPENSETR, RCC_MC_AHB2LPENCLRR	SR/CR	N	NS	-
RCC_MC_AHB3LPENSETR, RCC_MC_AHB3LPENCLRR	SR/CR	N	NS	-
RCC_MC_AHB4LPENSETR, RCC_MC_AHB4LPENCLRR	SR/CR	N	NS	-
RCC_MC_AHB5LPENSETR, RCC_MC_AHB5LPENCLRR	SR/CR	N	NS	-
RCC_MC_AXIMLPENSETR, RCC_MC_AXIMLPENCLRR	SR/CR	N	NS	-
RCC_MC_MLAHBLPENSETR, RCC_MC_MLAHBLPENCLRR	SR/CR	N	NS	-
RCC_VERR	R	N	NS	-
RCC_IDR	R	N	NS	-
RCC_SIDR	R	N	NS	-

1. **RW** means Read or Write.
   SR means Set or Read. On the register description it is noted “RWS1”.
   CR means Clear or Read.On the register description it is noted “RWC1”.
   R means Read.
   W means Write.
2. **N** means that there is no control on the master accessing this register.
   MPU Only means that only the MPU is allowed to write this resource. The read operations are always allowed.
3. **AS** means “Always Secure”: this register can only be accessed in secure mode.
   S means “Secure”: this register can only be written in secure mode when TZEN = ‘1’ or when TZEN AND MCKPROT = ‘1’. If TZEN = ‘0’, then the register can also be written in non-secure mode. Read access is always allowed.
   NS means “Non-secure”: this register always accept non-secure access, independently of TZEN bit. Secure and non-secure accesses are accepted.

10.7.1 Register mapping

All the RCC registers can be accessed in word, half-word and byte format, unless explicitly specified differently.

All the RCC registers are reset by the **nreset** signal, unless explicitly specified differently.

The RCC register map is divided into two sections:

- An address section (lower address part) containing registers which can be switched to secure mode,
- An address section (higher address part) containing registers which cannot be switched to secure mode.

Note that the burst access mode is not supported.

10.7.2 RCC TrustZone control register (RCC_TZCR)

Address offset: 0x000

Reset value: 0x0000 0003

This register is used to switch the RCC into secure mode. This register can only be accessed in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKPROT	TZEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCKPROT**: Protection of the generation of mcuss_ck clock (secure) Enable

Set and reset by software in order to control the security for all registers involved in the generation of the mcuss_ck clock. This bit is only significant if TZEN = '1'.

When TZEN = '1', if MCKPROT = '1', then only secure accesses are allowed to change the register controlling the mcuss_ck clock.

In all other cases, non secure accesses are allowed to those registers.

Refer to *Section 10.4.14: RCC TrustZone function* for details.

0: The registers controlling mcuss_ck clock are not protected
1: The registers controlling mcuss_ck clock are protected, if TZEN = '1' (default after reset).

Bit 0 **TZEN**: RCC TrustZone (secure) Enable

Set and reset by software in order to control the TrustZone mode. When this bit is set, the control of several RCC functions shall be controlled using secure accesses. Refer to *Section 10.4.14: RCC TrustZone function* for details.

0: No protection
1: TrustZone enabled (default after reset).

10.7.3 RCC oscillator clock enable set register (RCC_OCENSETR)

Address offset: 0x00C

Reset value: 0x0000 0001

This register is used to control the oscillators.Writing '0' to this register has no effect, writing '1' will set the corresponding bits. Reading will give the effective values of each bit.If TZEN = MCKPROT = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSECSSON	HSEBYP	HSEKERON	HSEON	DIGBYP	Res.	CSIKERON	CSION	Res.	Res.	HSIKERON	HSION
				rs	rs	rs	rs	rs		rs	rs			rs	rs

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **HSECSSON**: Set the HSECSSON bit

Set the Clock Security System on HSE.

This bit is “set only” (disabled by a system reset or when the system enters in Standby mode).

When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready.

0: Reading '0' means that the Clock Security System on HSE is OFF (default after reset)

1: Writing '1' enables the Clock Security System on HSE, reading '1' means that the Clock Security System on HSE is ON

Bit 10 **HSEBYP**: Set HSEBYP bit

Set by software in order to enable the bypass mode.

0: No effect

1: Set the HSEBYP bit

Bit 9 **HSEKERON**: Set HSEKERON bit

Set by software to force the HSE to ON, even in (LP-)Stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of HSEON.

0: No effect

1: Set the HSEKERON bit

Bit 8 **HSEON**: Set HSEON bit

Set by software to enable the HSE.

0: No effect

1: Set HSEON bit

Bit 7 **DIGBYP**: Set DIGBYP bit

Set by software when the external clock connected to OSC_IN is a full-swing digital signal.

0: No effect

1: Set DIGBYP bit (digital bypass)

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CSIKERON**: Set CSIKERON bit

Set by software to force the CSI to ON, even in (LP-)Stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.

0: No effect
1: Set the CSIKERON bit

Bit 4 **CSION**: Set CSION bit

Set by software to enable CSI clock.

0: No effect
1: Set the CSION bit

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **HSIKERON**: Set HSIKERON bit

Set by software to force the HSI to ON, even in (LP-)Stop mode, in order to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.

0: No effect
1: Set the HSIKERON bit

Bit 0 **HSION**: Set HSION bit

Set by software to enable the HSI clock.

The HSION is also set by hardware to force the HSI to ON when the product leaves Standby mode or one of the Stop modes.

0: No effect
1: Set the HSION bit

10.7.4 RCC oscillator clock enable clear register (RCC_OCENCLRR)

Address offset: 0x010

Reset value: 0x0000 0001

This register is used to control the oscillators.Writing '0' to this register has no effect, writing '1' will clear the corresponding bits. Reading will give the effective values of the enable bits.If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSEBYP	HSEKERON	HSEON	DIGBYP	Res.	CSIKERON	CSION	Res.	Res.	HSIKERON	HSION
					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1			rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSEBYP**: Clear the HSEBYP bit

Cleared by software in order to disable the bypass mode.

0: No effect

1: Clear the HSEBYP bit

Bit 9 **HSEKERON**: Clear HSEKERON bit

Cleared by software in order to allow the HSE to be switched off in one of the Stop modes.

0: No effect

1: Clear the HSEKERON bit

Bit 8 **HSEON**: Clear of HSEON bit

Cleared by software to disable the HSE clock.

0: No effect

1: Clear the HSEON bit

Bit 7 **DIGBYP**: Clear of DIGBYP bit

Cleared by software when the external clock connected to OSC_IN is a low-swing signal.

0: No effect

1: Clear the DIGBYP bit (analog bypass)

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CSIKERON**: Clear of CSIKERON bit

Cleared by software in order to allow the CSI to be switched off in one of the Stop modes.

This bit has no effect on the value of CSION.

0: No effect

1: Clear the CSIKERON bit

Bit 4 **CSION**: Clear of CSION bit

Cleared by software to disable the CSI clock.

0: CSI is OFF

1: Clear the CSION bit

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **HSIKERON**: Clear of HSIKERON bit

Cleared by software in order to allow the HSI to be switched off in one of the Stop modes.

This bit has no effect on the value of HSION.

0: No effect

1: Clear the HSIKERON bit

Bit 0 **HSION**: Clear of HSION bit

Cleared by software to disable the HSI clock.

0: No effect

1: Clear the HSION bit

10.7.5 RCC oscillator clock ready register (RCC_OCRDYR)

Address offset: 0x808

Reset value: 0x0000 0000

This is a read-only access register, It contains the status flags of oscillators. Writing has no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CKREST	AXICKRDY	MPUCKRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						r	r	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDY	Res.	Res.	Res.	CSIRDY	Res.	HSIDIVRDY	Res.	HSIRDY
							r				r		r		r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CKREST**: Clock Restore State Machine Status

Set by hardware to indicate if the clock restore sequence is on-going.

0: The clock restore process is not on-going (default after reset)
1: The clock restore process is on-going

Bit 24 **AXICKRDY**: AXI sub-system clock ready flag

Set by hardware to indicate that the axiss_ck clock is available.

0: axiss_ck clock is not available (default after reset)
1: axiss_ck clock is available

Bit 23 **MPUCKRDY**: MPU clock ready flag

Set by hardware to indicate that the mpuss_ck clock is available.

0: mpuss_ck clock is not available (default after reset)
1: mpuss_ck clock is available

Bits 22:9 Reserved, must be kept at reset value.

Bit 8 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.

0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CSIRDY**: CSI clock ready flag

Set by hardware to indicate that the CSI oscillator is stable.

0: CSI clock is not ready (default after reset)
1: CSI clock is ready

Bit 3 Reserved, must be kept at reset value.

Bit 2 **HSIDIVRDY**: HSI divider ready flag

Set and reset by hardware.

As a write in HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF will go immediately to '0' when HSIDIV value is changed, and will be set back to '1' when the output frequency matches the value programmed into HSIDIV

0: the new division ratio is not yet propagated to hsi_ck (hsi_ker_ck) (default after reset)
1: the hsi_ck (hsi_ker_ck) clock frequency reflects the new HSIDIV value

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HSIRDY**: HSI clock ready flag

Set by hardware to indicate that the HSI oscillator is stable.

0: HSI clock is not ready (default after reset)
1: HSI clock is ready

10.7.6 RCC debug configuration register (RCC_DBGCFGR)

Address offset: 0x80C

Reset value: 0x0000 0001

This is register contains the enable control of the debug and trace function, and the clock divider for the trace function.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBGRST	Res.	Res.	TRACECKEN	DBGCKEN	Res.	Res.	Res.	Res.	Res.	TRACEDIV[2:0]		
			rw			rw	rw						rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **DBGRST**: Reset of the debug function

Set and cleared by software to control the generation of the reset (dbg_rstn) for the trace and debug parts.

0: The trace and debug parts are not reset. (default after reset)

1: The trace and debug parts are under reset.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **TRACECKEN**: Clock enable for trace function

Set and cleared by software to control the generation of the trace_ck clock.

See *Section : Clock distribution for DBG and trace* for additional information.

0: The clock for the trace function is disabled (default after reset)

1: The clock for the trace function is enabled

Bit 8 **DBGCKEN**: Clock enable for debug function

Set and cleared by software to control the generation of the ck_apb_dbg clock (DEBUG APB clock), the ck_sys_dbg. See *Section : Clock distribution for DBG and trace* for additional information.

0: The enabling of the clock for the debug function is controlled by cdbgwrupreq signal from DAP. (default after reset)

1: The clock for the debug function is enabled

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TRACEDIV[2:0]**: Clock divider for the trace clock (ck_trace)

Set and reset by software to control the ck_trace clock division factor.

It is possible to change this division ratio on-the-fly.

others: aclk / 16

0x0: aclk

0x1: aclk / 2 (default after reset)

0x2: aclk / 4

0x3: aclk / 8

10.7.7 RCC HSI configuration register (RCC_HSICFGR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to configure the HSI. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HSICAL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	HSIDIV[1:0]	
	rw	rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HSICAL[11:0]**: HSI clock calibration

Set by hardware by option byte loading during system reset

Adjusted by software through trimming bits HSITRIM.

This field represents the sum of engineering Option Byte calibration value and HSITRIM bits value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **HSITRIM[6:0]**: HSI clock trimming

Set by software to adjust calibration. HSITRIM represents a signed value.

HSITRIM field is added to the engineering Option Bytes loaded during reset phase (bsec_hsi_cal[11:0]) in order to form the calibration trimming value.

HSICAL[11:0] = HSITRIM[6:0] + bsec_hsi_cal[11:0].

0x40: bsec_hsi_cal[11:0] - 64
0x41: bsec_hsi_cal[11:0] - 63
...
0x0: bsec_hsi_cal[11:0] (default after reset)
...
0x3E: bsec_hsi_cal[11:0] + 62
0x3F: bsec_hsi_cal[11:0] + 63

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **HSIDIV[1:0]**: HSI clock divider

Set and reset by software.

These bits allows the user to select a division ratio in order to select the wanted HSI clock frequency. The bit HSIDIVRDY of *RCC oscillator clock ready register (RCC_OCRDYR)* indicates when the new division ratio is taken into account.

0x0: Division by 1, hsi_ck (hsi_ker_ck) = 64 MHz (default after reset)
0x1: Division by 2, hsi_ck (hsi_ker_ck) = 32 MHz
0x2: Division by 4, hsi_ck (hsi_ker_ck) = 16 MHz
0x3: Division by 8, hsi_ck (hsi_ker_ck) = 8 MHz

10.7.8 RCC CSI configuration register (RCC_CSICFGR)

Address offset: 0x01C

Reset value: 0x0000 1000

This register is used to fine-tune the CSI frequency. If TZEN = MCKPROT = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSICAL[7:0]							
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CSITRIM[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CSICAL[7:0]**: CSI clock calibration
Set by hardware by option byte loading during system reset.
Adjusted by software through trimming bits CSITRIM.
This field represents the sum of engineering Option Byte calibration value and CSITRIM bits value

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **CSITRIM[4:0]**: CSI clock trimming
Set by software to adjust calibration.
CSITRIM field is added to the engineering Option Bytes loaded during reset phase (bsec_csi_cal[7:0]) in order to form the calibration trimming value.
CSICAL[7:0] = CSITRIM[4:0] + bsec_csi_cal[7:0].

Bits 7:0 Reserved, must be kept at reset value.

10.7.9 RCC MCO1 configuration register (RCC_MCO1CFGR)

Address offset: 0x800

Reset value: 0x0000 0000

This register is used to select the clock generated on MCO1 output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	MCO1ON	Res.	Res.	Res.	Res.	MCO1DIV[3:0]				Res.	MCO1SEL[2:0]		
			rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **MCO1ON**: Control of the MCO1 output

Set and cleared by software to enable the MCO1 output

0: The MCO1 output is disabled
1: The MCO1 output is enabled

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **MCO1DIV[3:0]**: MCO1 prescaler

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

Refer to *Section 10.4.4: Clock output generation (MCO1 and MCO2)* for details.

0x0: bypass (default after reset)
0x1: division by 2
0x2: division by 3
...
0xF: division by 16

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **MCO1SEL[2:0]**: MCO1 clock output selection

Set and cleared by software. Clock source selection may generate glitches on MCO1.

It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.

0x0: HSI clock selected (hsi_ck) (default after reset)
0x1: HSE clock selected (hse_ck)
0x2: CSI clock selected (csi_ck)
0x3: LSI clock selected (lsi_ck)
0x4: LSE oscillator clock selected (lse_ck)
others: reserved

10.7.10 RCC MCO2 configuration register (RCC_MCO2CFGR)

Address offset: 0x804

Reset value: 0x0000 0000

This register is used to select the clock generated on MCO2 output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	MCO2ON	Res.	Res.	Res.	Res.	MCO2DIV[3:0]				Res.	MCO2SEL[2:0]		
			rw					rw	rw	rw	rw		rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **MCO2ON**: Control of the MCO2 output

Set and cleared by software to enable the MCO2 output

0: The MCO2 output is disabled

1: The MCO2 output is enabled

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **MCO2DIV[3:0]**: MCO2 prescaler

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.

Refer to *Section 10.4.4: Clock output generation (MCO1 and MCO2)* for details.

0x0: bypass (default after reset)

0x1: division by 2

0x2: division by 3

...

0xF: division by 16

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **MCO2SEL[2:0]**: Micro-controller clock output 2

Set and cleared by software. Clock source selection may generate glitches on MCO2.

It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.

0x0: MPU clock selected (mpuss_ck) (default after reset)

0x1: AXI clock selected (axiss_ck)

0x2: MCU clock selected (mcuss_ck)

0x3: PLL4 clock selected (pll4_p_ck)

0x4: HSE clock selected (hse_ck)

0x5: HSI clock selected (hsi_ck)

others: reserved

10.7.11 RCC MPU clock selection register (RCC_MPCKSELR)

Address offset: 0x020

Reset value: 0x8000 0000

This register is used to select the clock source for the MPU. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPUSRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUSRC[1:0]	
														rw	rw

Bit 31 **MPUSRCRDY**: MPU clock switch status

Set and reset by hardware to indicate if the MPU clock switch transition has been properly performed.

0: The MPU switch is not ready: no clock is generated on its output
1: The MPU switch is ready: the clock switch is selecting the clock given by MPUSRC field. (default after reset)

Bits 30:2 Reserved, must be kept at reset value.

Bits 1:0 **MPUSRC[1:0]**: MPU clock switch

Set and reset by software to select the MPU sub-system clock source (mpuss_ck).

Note that MPUDIV is disabled if MPUSRC[1:0] is different from '11'.

Be sure that MPUDIV is different from '000' before setting MPUSRC to '11'.

0x0: HSI selected as MPU sub-system clock (hsi_ck) (default after reset)
0x1: HSE selected as MPU sub-system clock (hse_ck)
0x2: PLL1 selected as MPU sub-system clock (pll1_p_ck)
0x3: PLL1 via MPUDIV is selected as MPU sub-system clock (pll1_p_ck / 2 MPUDIV).

10.7.12 RCC AXI sub-system clock selection register (RCC_ASSCKSELR)

Address offset: 0x024

Reset value: 0x8000 0000

This register is used to select the clock source for the AXI sub-system. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AXISSRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXISSRC[2:0]		
													rw	rw	rw

Bit 31 **AXISSRCRDY**: AXI sub-system clock switch status

Set and reset by hardware to indicate if the AXI clock switch transition has been properly performed.

0: The AXI sub-system switch is not ready or in positions higher than '011': no clock is generated on its output

1: The AXI sub-system switch is ready: the clock switch is selecting the clock given by AXISSRC field. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **AXISSRC[2:0]**: AXI sub-system clock switch

Set and reset by software to select the AXI sub-system clock source (axiss_ck).

0x0: HSI selected as AXI sub-system clock (hsi_ck) (default after reset)

0x1: HSE selected as AXI sub-system clock (hse_ck)

0x2: PLL2 selected as AXI sub-system clock (pll2_p_ck)

others: axiss_ck is gated

10.7.13 RCC MCU sub-system clock selection register (RCC_MSSCKSELR)

Address offset: 0x048

Reset value: 0x8000 0000

This register is used to select the clock source for the MCU sub-system, including the MCU itself. If TZEN = MCKPROT = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCUSSRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCUSSRC[1:0]	
														rw	rw

Bit 31 **MCUSSRCRDY**: MCU sub-system clock switch status

Set and reset by hardware to indicate if the MCU clock switch transition has been properly performed.

0: The MCU sub-system switch is not ready or in positions higher than '011': no clock is generated on its output

1: The MCU sub-system switch is ready: the clock switch is selecting the clock given by MCUSSRC field. (default after reset)

Bits 30:2 Reserved, must be kept at reset value.

Bits 1:0 **MCUSSRC[1:0]**: MCUSS clock switch

Set and reset by software to select the MCU sub-system clock source (mcuss_ck).

Reset by hardware in order to select hsi_ck when the system exits from one of the Stop modes.

0x0: HSI selected as MCU sub-system clock (hsi_ck) (default after reset)

0x1: HSE selected as MCU sub-system clock (hse_ck)

0x2: CSI selected as MCU sub-system clock (csi_ck)

0x3: PLL3 selected as MCU sub-system clock (pll3_p_ck)

10.7.14 RCC PLL 1 and 2 reference clock selection register (RCC_RCK12SELR)

Address offset: 0x028

Reset value: 0x8000 0000

This register is used to select the reference clock for PLL1 and PLL2. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL12SRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL12SRC[1:0]	
														rw	rw

Bit 31 **PLL12SRCRDY**: PLL12 reference clock switch status

Set and reset by hardware to indicate if the PLL12 clock switch transition has been properly performed.

0: The PLL12 switch is not ready or in position '1x': no clock is generated on its output

1: The PLL12 switch is ready: the clock switch is selecting the clock given by PLL12SRC field. (default after reset)

Bits 30:2 Reserved, must be kept at reset value.

Bits 1:0 **PLL12SRC[1:0]**: Reference clock selection for PLL1 and PLL2

Set and reset by software to select the PLL1 and PLL2 clock source.

In order to save power when PLL1 and PLL2 are not used, PLL12SRC must be set to '1x'.

0x0: HSI selected as PLL clock (hsi_ck) (default after reset)

0x1: HSE selected as PLL clock (hse_ck)

others: No clock send to DIVMx divider and PLLs

10.7.15 RCC PLL 3 reference clock selection register (RCC_RCK3SELR)

Address offset: 0x820

Reset value: 0x8000 0000

This register is used to select the reference clock for PLL3. If TZEN = MCKPROT = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL3SRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3SRC[1:0]	
														rw	rw

Bit 31 **PLL3SRCRDY**: PLL3 reference clock switch status

Set and reset by hardware to indicate if the PLL3 clock switch transition has been properly performed.

0: The PLL3 switch is not ready or in position '11': no clock is generated on its output

1: The PLL3 switch is ready: the clock switch is selecting the clock given by PLL3SRC field. (default after reset)

Bits 30:2 Reserved, must be kept at reset value.

Bits 1:0 **PLL3SRC[1:0]**: Reference clock selection for PLL3

Set and reset by software to select the PLL3 clock source.

In order to save power, when PLL3 is not used, PLL3SRC must be set to '11'.

0x0: HSI selected as PLL clock (hsi_ck) (default after reset)

0x1: HSE selected as PLL clock (hse_ck)

0x2: CSI selected as PLL clock (csi_ck)

0x3: No clock send to DIVMx divider and PLLs

10.7.16 RCC PLL4 reference clock selection register (RCC_RCK4SELR)

Address offset: 0x824

Reset value: 0x8000 0000

This register is used to select the reference clock for PLL4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL4SRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4SRC[1:0]	
														rw	rw

Bit 31 **PLL4SRCRDY**: PLL4 reference clock switch status

Set and reset by hardware to indicate if the PLL4 clock switch transition has been properly performed.

0: The PLL4 switch is not ready or in position '11': no clock is generated on its output

1: The PLL4 switch is ready: the clock switch is selecting the clock given by PLL4SRC field. (default after reset)

Bits 30:2 Reserved, must be kept at reset value.

Bits 1:0 **PLL4SRC[1:0]**: Reference clock selection for PLL4

Set and reset by software to select the PLL4 clock source.

0x0: HSI selected as PLL clock (hsi_ck) (default after reset)

0x1: HSE selected as PLL clock (hse_ck)

0x2: CSI selected as PLL clock (csi_ck)

0x3: Signal I2S_CKIN used as reference clock

10.7.17 RCC TIM group 1 prescaler register (RCC_TIMG1PRER)

Address offset: 0x828

Reset value: 0x8000 0000

This register is used to control the prescaler value of timers located into APB1 domain. It concerns TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, TIM12, TIM13 and TIM14. Refer to *Section : Sub-system clock generation* for additional information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMG1PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG1PRE
															rw

Bit 31 **TIMG1PRERDY**: Timers clocks prescaler status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account (default after reset).

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **TIMG1PRE**: Timers clocks prescaler selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 domain.

Refer to *Table 58: Ratio between clock timers and pclk*.

0: The Timers kernel clock is equal to mlhclk if APB1DIV is corresponding to a division by 1 or 2, else it is equal to 2 x Fpclk1 (default after reset)

1: The Timers kernel clock is equal to mlhclk if APB1DIV is corresponding to division by 1, 2 or 4, else it is equal to 4 x Fpclk1

10.7.18 RCC TIM group 2 prescaler register (RCC_TIMG2PRER)

Address offset: 0x82C

Reset value: 0x8000 0000

This register is used to control the prescaler value of timers located into APB2 domain. It concerns TIM1, TIM8, TIM15, TIM16, and TIM17. Refer to *Section : Sub-system clock generation* for additional information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMG2PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG2PRE
															rw

Bit 31 **TIMG2PRERDY**: Timers clocks prescaler status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account (default after reset).

Bits 30:1 Reserved, must be kept at reset value.

Bit 0 **TIMG2PRE**: Timers clocks prescaler selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB2 domain.

Refer to *Table 58: Ratio between clock timers and pclk*.

0: The Timers kernel clock is equal to mlhclk if APB2DIV is corresponding to a division by 1 or 2, else it is equal to 2 x Fpclk2 (default after reset)

1: The Timers kernel clock is equal to mlhclk if APB2DIV is corresponding to division by 1, 2 or 4, else it is equal to 4 x Fpclk2

10.7.19 RCC RTC clock division register (RCC_RTCDIVR)

Address offset: 0x044

Reset value: 0x0000 0000

This register is used to divide the HSE clock for RTC. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDIV[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RTCDIV[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE to generate a clock for RTC.

Caution: The software has to set these bits correctly to ensure that the clock supplied to the RTC is lower than 4 MHz. These bits must be configured if needed before selecting the RTC clock source.

Note that when the RTCSRC is different from 3, this divider is disabled (does not provide clock).

0x0: HSE (default after reset)
0x1: HSE/2
0x2: HSE/3
0x3: HSE/4
...
0x3E: HSE/63
0x3F: HSE/64

10.7.20 RCC MPU clock divider register (RCC_MPCKDIVR)

Address offset: 0x02C

Reset value: 0x8000 0001

This register is used to control the MPU clock prescaler. Refer to *Section : Sub-system clock generation* for additional information. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPUDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUDIV[2:0]		
													rw	rw	rw

Bit 31 **MPUDIVRDY**: MPU sub-system clock divider status

Set and reset by hardware to indicate if the new division factor is taken into account.

This bit significant only when MPUSRC[1:0] = '11'.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **MPUDIV[2:0]**: MPU Core clock divider

Set and reset by software to control the MPU clock division factor when the MPUSRC is set to position 3.

It is possible to change this division ratio on-the-fly. It impacts only the frequency of the MPU clock. The clocks are divided with the new prescaler factor, from 1 to 16 pll1_p_ck cycles after MPUDIV update. The application can check if the new division factor is taken into account by reading back the MPUDIVRDY flag. Note that this divider is disabled if MPUSRC[1:0] is different from '11'.

0x0: The MPUDIV is disabled; i.e. no clock generated

0x1: The mpuss_ck is equal to pll1_p_ck divided by 2 (default after reset)

0x2: The mpuss_ck is equal to pll1_p_ck divided by 4

0x3: The mpuss_ck is equal to pll1_p_ck divided by 8

others: The mpuss_ck is equal to pll1_p_ck divided by 16

10.7.21 RCC AXI clock divider register (RCC_AXIDIVR)

Address offset: 0x030

Reset value: 0x8000 0000

This register is used to control the AXI Matrix clock prescaler. Refer to *Section : Sub-system clock generation* for additional information. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AXIDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIDIV[2:0]		
													rw	rw	rw

Bit 31 **AXIDIVRDY**: AXI sub-system clock divider status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.
1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **AXIDIV[2:0]**: AXI, AHB5 and AHB6 clock divider

Set and reset by software to control the AXI, AHB5 and AHB6 clock division factor.

It is possible to change this division ratio on-the-fly. It impacts the frequency of AXI, APB4, APB5 AHB5 and AHB6 clocks.

The clocks are divided with the new prescaler factor, from 1 to 4 axiss_ck cycles after AXIDIV update. The application can check if the new division factor is taken into account by reading back the AXIDIVRDY flag.

0x0: axiss_ck (default after reset)
0x1: axiss_ck / 2
0x2: axiss_ck / 3
others: axiss_ck / 4

10.7.22 RCC APB4 clock divider register (RCC_APB4DIVR)

Address offset: 0x03C

Reset value: 0x8000 0000

This register is used to control the APB4 clock divider. Refer to *Section : Sub-system clock generation* for additional information. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB4DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB4DIV[2:0]		
													rw	rw	rw

Bit 31 **APB4DIVRDY**: APB4 clock divider status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB4DIV[2:0]**: APB4 clock divider

Set and reset by software to control the APB4 clock division factor.

It is possible to change this division ratio on-the-fly. It impacts only the frequency of APB4 clock.

The clocks are divided with the new prescaler factor, from 1 to 16 aclk cycles after APB4DIV update. The application can check if the new division factor is taken into account by reading back the APB4DIVRDY flag.

0x0: aclk (default after reset)

0x1: aclk / 2

0x2: aclk / 4

0x3: aclk / 8

others: aclk / 16

10.7.23 RCC APB5 clock divider register (RCC_APB5DIVR)

Address offset: 0x040

Reset value: 0x8000 0000

This register is used to control the APB5 clock divider. Refer to *Section : Sub-system clock generation* for additional information. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB5DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB5DIV[2:0]		
													rw	rw	rw

Bit 31 **APB5DIVRDY**: APB5 clock divider status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB5DIV[2:0]**: APB5 clock divider

Set and reset by software to control the APB5 clock division factor.

It is possible to change this division ratio on-the-fly. It impacts only the frequency of APB5 clock.

The clocks are divided with the new prescaler factor, from 1 to 16 aclk cycles after APB5DIV update. The application can check if the new division factor is taken into account by reading the APB5DIVRDY flag.

0x0: aclk (default after reset)

0x1: aclk / 2

0x2: aclk / 4

0x3: aclk / 8

others: aclk / 16

10.7.24 RCC MCU clock divider register (RCC_MCUDIVR)

Address offset: 0x830

Reset value: 0x8000 0000

This register is used to control the MCU sub-system clock prescaler. Refer to *Section : Sub-system clock generation* for additional information. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCUDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCUDIV[3:0]			
												rw	rw	rw	rw

Bit 31 **MCUDIVRDY**: MCU clock prescaler status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.
1: The new division factor is taken into account. (default after reset)

Bits 30:4 Reserved, must be kept at reset value.

Bits 3:0 **MCUDIV[3:0]**: MCU clock divider

Set and reset by software to control the MCU clock division factor.

Changing this division ratio has an impact on the frequency of MCU clock, and all bus matrix clocks. The clocks are divided with the new prescaler factor, from 1 to 512 cycles of mcuss_ck, after MCUDIV update. The application can check if the new division factor is taken into account by reading back the MCUDIVRDY flag.

0x0: mcuss_ck not divided (default after reset)
0x1: mcuss_ck divided by 2
0x2: mcuss_ck divided by 4
0x3: mcuss_ck divided by 8
0x4: mcuss_ck divided by 16
0x5: mcuss_ck divided by 32
0x6: mcuss_ck divided by 64
0x7: mcuss_ck divided by 128
0x8: mcuss_ck divided by 256
others: mcuss_ck divided by 512

10.7.25 RCC APB1 clock divider register (RCC_APB1DIVR)

Address offset: 0x834

Reset value: 0x8000 0000

This register is used to control the APB1 clock prescaler. Refer to *Section : Sub-system clock generation* for additional information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB1DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB1DIV[2:0]		
													rw	rw	rw

Bit 31 **APB1DIVRDY**: APB1 clock prescaler status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB1DIV[2:0]**: APB1 clock divider

Set and reset by software to control the clock division factor.

The clocks are divided with the new prescaler factor, from 1 to 16 mlhclk cycles after APB1DIV update. The application can check if the new division factor is taken into account by reading back the APB1DIVRDY flag.

0x0: mlhclk (default after reset)
0x1: mlhclk / 2
0x2: mlhclk / 4
0x3: mlhclk / 8
0x4: mlhclk / 16
others: Not allowed

10.7.26 RCC APB2 clock divider register (RCC_APB2DIVR)

Address offset: 0x838

Reset value: 0x8000 0000

This register is used to control the APB2 clock prescaler. Refer to *Section : Sub-system clock generation* for additional information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB2DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB2DIV[2:0]		
													rw	rw	rw

Bit 31 **APB2DIVRDY**: APB2 clock prescaler status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB2DIV[2:0]**: APB2 clock divider

Set and reset by software to control the clock division factor.

The clocks are divided with the new prescaler factor, from 1 to 16 mlhclk cycles after APB2DIV update. The application can check if the new division factor is taken into account by reading back the APB2DIVRDY flag.

0x0: mlhclk (default after reset)

0x1: mlhclk / 2

0x2: mlhclk / 4

0x3: mlhclk / 8

0x4: mlhclk / 16

others: Not allowed

10.7.27 RCC APB3 clock divider register (RCC_APB3DIVR)

Address offset: 0x83C

Reset value: 0x8000 0000

This register is used to control the APB3 clock prescaler. Refer to *Section : Sub-system clock generation* for additional information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APB3DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIV[2:0]		
													rw	rw	rw

Bit 31 **APB3DIVRDY**: APB3 clock prescaler status

Set and reset by hardware to indicate if the new division factor is taken into account.

0: The new division factor is not yet taken into account.

1: The new division factor is taken into account. (default after reset)

Bits 30:3 Reserved, must be kept at reset value.

Bits 2:0 **APB3DIV[2:0]**: APB3 clock divider

Set and reset by software to control the clock division factor.

The clocks are divided with the new prescaler factor, from 1 to 16 hclk cycles after APB3DIV update. The application can check if the new division factor is taken into account by reading back the APB3DIVRDY flag.

0x0: hclk (default after reset)

0x1: hclk / 2

0x2: hclk / 4

0x3: hclk / 8

others: hclk / 16

10.7.28 RCC PLL1 control register (RCC_PLL1CR)

Address offset: 0x080

Reset value: 0x0000 0000

This register is used to control the PLL1. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVREN	DIVQEN	DIVPEN	Res.	SSCG_CTRL	PLL1RDY	PLLON
									rw	rw	rw		rw	r	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DIVREN**: PLL1 DIVR divider output enable
Set and reset by software to enable the pll1_r_ck output of the PLL1.
In order to save power, when the pll1_r_ck is not needed, DIVREN and DIVR must be set to '0'.
0: pll1_r_ck output is disabled (default after reset)
1: pll1_r_ck output is enabled

Bit 5 **DIVQEN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the pll1_q_ck output of the PLL1.
In order to save power, when the pll1_q_ck is not needed, DIVQEN and DIVQ must be set to '0'.
0: pll1_q_ck output is disabled (default after reset)
1: pll1_q_ck output is enabled

Bit 4 **DIVPEN**: PLL1 DIVP divider output enable
Set and reset by software to enable the pll1_p_ck output of the PLL1.
In order to save power, when the pll1_p_ck is not needed, DIVPEN and DIVP must be set to '0'.
0: pll1_p_ck output is disabled (default after reset)
1: pll1_p_ck output is enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSCG_CTRL**: Spread Spectrum Clock Generator of PLL1 enable

Set and reset by software to enable the spread spectrum clock generator of PLL1, in order to reduce the amount of EMI peaks.

0: Clock Spreading Generator disabled (default after reset)

1: Clock Spreading Generator enabled

Bit 1 **PLL1RDY**: PLL1 clock ready flag

Set by hardware to indicate that the PLL1 is locked.

0: PLL1 unlocked (default after reset)

1: PLL1 locked

Bit 0 **PLLON**: PLL1 enable

Set and cleared by software to enable the PLL1.

Note that DIVPEN of PLL1 must be set to ‘0’ before setting PLLON to ‘0’. Please refer to *Section : PLL disabling procedure* for details.

0: PLL1 OFF (default after reset)

1: PLL1 is ON, and ref1_ck is provided to the PLL1

10.7.29 RCC PLL1 configuration register 1 (RCC_PLL1CFGR1)

Address offset: 0x084

Reset value: 0x0001 0031

This register is used to configure the PLL1. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVM1[5:0]					
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVN[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **DIVM1[5:0]**: Prescaler for PLL1
Set and cleared by software to configure the prescaler of the PLL1.
0x0: bypass
0x1: division by 2 (default after reset)
0x2: division by 3
...
0x3F: division by 64

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DIVN[8:0]**: Multiplication factor for PLL1 VCO
Set and reset by software to control the multiplication factor of the VCO.
Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is: 800 to 1600 MHz.
VCO output frequency = Fref1_ck x 2 x (DIVN+1), when value 0 has been loaded into FRACV, with:
- Valid division ratios for DIVN: between 25 and 100
- The input frequency Fref1_ck between 8 and 16 MHz
0x18: Division ratio is 25
0x19: Division ratio is 26
...
0x31: Division ratio is 50 (default after reset)
...
0x63: Division ratio is 100
Others: wrong configurations

10.7.30 RCC PLL1 configuration register 2 (RCC_PLL1CFGR2)

Address offset: 0x088

Reset value: 0x0001 0100

This register is used to configure the PLL1. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DIVQ[6:0]							Res.	DIVP[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **DIVR[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the pll1_r_ck clock.

0x0: pll1_r_ck = fout1_ck
0x1: pll1_r_ck = fout1_ck / 2 (default after reset)
0x2: pll1_r_ck = fout1_ck / 3
...
0x7F: pll1_r_ck = fout1_ck / 128

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **DIVQ[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the pll1_q_ck clock.

0x0: pll1_q_ck = fout1_ck
0x1: pll1_q_ck = fout1_ck / 2 (default after reset)
0x2: pll1_q_ck = fout1_ck / 3
...
0x7F: pll1_q_ck = fout1_ck / 128

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DIVP[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the pll1_p_ck clock.

Odd division factors are not recommended due to duty cycle degradation.

0x0: pll1_p_ck = fout1_ck (default after reset)
0x1: pll1_p_ck = fout1_ck / 2
0x2: pll1_p_ck = fout1_ck / 3
...
0x7F: pll1_p_ck = fout1_ck / 128

10.7.31 RCC PLL1 fractional register (RCC_PLL1FRACR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register is used to fine-tune the frequency of the PLL1 VCO. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACLE
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRACV[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **FRACLE**: PLL1 fractional latch enable

Set and reset by software to latch the content of FRACV into the Sigma-Delta modulator.

In order to latch the FRACV value into the Sigma-Delta modulator, FRACLE must be set to '0', then set to '1': the transition 0 to '1' transfers the content of FRACV into the modulator. Refer to *Section : Reprogramming post-dividers* and *Section : PLL initialization procedure* for additional information on the PLL programming.

Bits 15:3 **FRACV[12:0]**: Fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is: 800 to 1600 MHz

VCO output frequency = Fref1_ck x ((DIVN+1) + (FRACV / 8192)), with

- DIVN shall be between 4 and 512
- FRACV can be between 0 and 8191
- The input frequency Fref1_ck shall be between 4 and 16 MHz

In order to change the FRACV value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit FRACLE to '0',
- write the new fractional value into FRACV,
- set the bit FRACLE to '1'.

Bits 2:0 Reserved, must be kept at reset value.

10.7.32 RCC PLL1 clock spreading generator register (RCC_PLL1CSGR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is used to configure the PLL1.It is not recommended to change the content of this register when the PLL1 is enabled (PLLON = '1'). Refer to *Section : Using the PLLs in spread spectrum mode* for details. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	INC_STEP[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSCG_MODE	RPDFN_DIS	TPDFN_DIS	MOD_PER[12:0]												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **INC_STEP[14:0]**: Modulation Depth Adjustment for PLL1
Set and reset by software to adjust the modulation depth of the clock spreading generator.

Bit 15 **SSCG_MODE**: Spread spectrum clock generator mode
Set and reset by software to select the clock spreading mode.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bit 14 **RPDFN_DIS**: Dithering RPDF noise control
Set and reset by software.
This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.
0: Dithering noise injection enabled (default after reset)
1: Dithering noise injection disabled

Bit 13 **TPDFN_DIS**: Dithering TPDF noise control
Set and reset by software.
This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.
0: Dithering noise injection enabled (default after reset)
1: Dithering noise injection disabled

Bits 12:0 **MOD_PER[12:0]**: Modulation Period Adjustment for PLL1
Set and reset by software to adjust the modulation period of the clock spreading generator.

10.7.33 RCC PLL2 control register (RCC_PLL2CR)

Address offset: 0x094

Reset value: 0x0000 0000

This register is used to control the PLL2. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVREN	DIVQEN	DIVPEN	Res.	SSCG_CTRL	PLL2RDY	PLLON
									rw	rw	rw		rw	r	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DIVREN**: PLL2 DIVR divider output enable

Set and reset by software to enable the pll2_r_ck output of the PLL2.

In order to save power, when the pll2_r_ck is not needed, DIVREN and DIVR must be set to '0'.

0: pll2_r_ck output is disabled (default after reset)
1: pll2_r_ck output is enabled

Bit 5 **DIVQEN**: PLL2 DIVQ divider output enable

Set and reset by software to enable the pll2_q_ck output of the PLL2.

In order to save power, when the pll2_q_ck is not needed, DIVQEN and DIVQ must be set to '0'.

0: pll2_q_ck output is disabled (default after reset)
1: pll2_q_ck output is enabled

Bit 4 **DIVPEN**: PLL2 DIVP divider output enable

Set and reset by software to enable the pll2_p_ck output of the PLL2.

In order to save power, when the pll2_p_ck is not needed, DIVPEN and DIVP must be set to '0'.

0: pll2_p_ck output is disabled (default after reset)
1: pll2_p_ck output is enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSCG_CTRL**: Clock Spreading Generator of PLL2 enable

Set and reset by software to enable the clock spreading generator of PLL2, in order to reduce the amount of EMI peaks.

0: Clock Spreading Generator disabled (default after reset)
1: Clock Spreading Generator enabled

Bit 1 **PLL2RDY**: PLL2 clock ready flag

Set by hardware to indicate that the PLL2 is locked.

0: PLL2 unlocked (default after reset)
1: PLL2 locked

Bit 0 **PLLON**: PLL2 enable

Set and cleared by software to enable the PLL2.

0: PLL2 OFF (default after reset)
1: PLL2 ON, and ref2_ck is provided to the PLL2

10.7.34 RCC PLL2 configuration register 1 (RCC_PLL2CFGR1)

Address offset: 0x098

Reset value: 0x0001 0063

This register is used to configure the PLL2. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVM2[5:0]					
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVN[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **DIVM2[5:0]**: Prescaler for PLL2

Set and cleared by software to configure the prescaler of the PLL2.

0x0: bypass
0x1: division by 2 (default after reset)
0x2: division by 3
...
0x3F: division by 64

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DIVN[8:0]**: Multiplication factor for PLL2 VCO

Set and reset by software to control the multiplication factor of the VCO.

Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is: 800 to 1600 MHz.

VCO output frequency = Fref2_ck x (DIVN+1), when value 0 has been loaded into FRACV, with:

- Valid division ratios for DIVN: between 25 and 100
- The input frequency Fref2_ck between 8 and 16 MHz

Others: wrong configurations

0x18: Division ratio is 25
0x19: Division ratio is 26
...
0x31: Division ratio is 50
...
0x63: Division ratio is 100 (default after reset)

10.7.35 RCC PLL2 configuration register 2 (RCC_PLL2CFGR2)

Address offset: 0x09C

Reset value: 0x0001 0101

This register is used to configure the PLL2. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DIVQ[6:0]							Res.	DIVP[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **DIVR[6:0]**: PLL2 DIVR division factor

Set and reset by software to control the frequency of the pll2_r_ck clock.

0x0: pll2_r_ck = fout2_ck
0x1: pll2_r_ck = fout2_ck / 2 (default after reset)
0x2: pll2_r_ck = fout2_ck / 3
...
0x7F: pll2_r_ck = fout2_ck / 128

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **DIVQ[6:0]**: PLL2 DIVQ division factor

Set and reset by software to control the frequency of the pll2_q_ck clock.

0x0: pll2_q_ck = fout2_ck
0x1: pll2_q_ck = fout2_ck / 2 (default after reset)
0x2: pll2_q_ck = fout2_ck / 3
...
0x7F: pll2_q_ck = fout2_ck / 128

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DIVP[6:0]**: PLL2 DIVP division factor

Set and reset by software to control the frequency of the pll2_p_ck clock.

0x0: pll2_p_ck = fout2_ck
0x1: pll2_p_ck = fout2_ck / 2 (default after reset)
0x2: pll2_p_ck = fout2_ck / 3
...
0x7F: pll2_p_ck = fout2_ck / 128

10.7.36 RCC PLL2 fractional register (RCC_PLL2FRACR)

Address offset: 0x0A0

Reset value: 0x0000 0000

This register is used to fine-tune the frequency of the PLL2 VCO. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACLE
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRACV[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **FRACLE**: PLL2 fractional latch enable

Set and reset by software to latch the content of FRACV into the Sigma-Delta modulator.

In order to latch the FRACV value into the Sigma-Delta modulator, FRACLE must be set to '0', then set to '1': the transition 0 to '1' transfers the content of FRACV into the modulator. Refer to *Section : Reprogramming post-dividers* and *Section : PLL initialization procedure* for additional information on the PLL programming.

Bits 15:3 **FRACV[12:0]**: Fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is: 800 to 1600 MHz

VCO output frequency = Fref2_ck x ((DIVN+1) + (FRACV / 8192)), with

- DIVN shall be between 4 and 512
- FRACV can be between 0 and 8191
- The input frequency Fref2_ck shall be between 4 and 16 MHz

In order to change the FRACV value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit FRACLE to '0',
- write the new fractional value into FRACV,
- set the bit FRACLE to '1'.

Bits 2:0 Reserved, must be kept at reset value.

10.7.37 RCC PLL2 clock spreading generator register (RCC_PLL2CSGR)

Address offset: 0x0A4

Reset value: 0x0000 0000

This register is used to configure the PLL2. It is not recommended to change the content of this register when the PLL2 is enabled (PLLON = '1'). Refer to *Section : Using the PLLs in spread spectrum mode* for details. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	INC_STEP[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSCG_MODE	RPDFN_DIS	TPDFN_DIS	MOD_PER[12:0]												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **INC_STEP[14:0]**: Modulation Depth Adjustment for PLL2
Set and reset by software to adjust the modulation depth of the clock spreading generator.

Bit 15 **SSCG_MODE**: Spread spectrum clock generator mode
Set and reset by software to select the clock spreading mode.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bit 14 **RPDFN_DIS**: Dithering RPDF noise control
Set and reset by software.
This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.
0: Dithering noise injection enabled (default after reset)
1: Dithering noise injection disabled

Bit 13 **TPDFN_DIS**: Dithering TPDF noise control
Set and reset by software.
This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.
0: Dithering noise injection enabled (default after reset)
1: Dithering noise injection disabled

Bits 12:0 **MOD_PER[12:0]**: Modulation Period Adjustment for PLL2
Set and reset by software to adjust the modulation period of the clock spreading generator.

10.7.38 RCC PLL3 control register (RCC_PLL3CR)

Address offset: 0x880

Reset value: 0x0000 0000

This register is used to control the PLL3. If TZEN = MCKPROT = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVREN	DIVQEN	DIVPEN	Res.	Res.	PLL3RDY	PLLON
									rw	rw	rw			r	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DIVREN**: PLL3 DIVR divider output enable

Set and reset by software to enable the pll3_r_ck output of the PLL3.

In order to save power, when the pll3_r_ck is not needed, DIVREN and DIVR must be set to '0'.

0: pll3_r_ck output is disabled (default after reset)
1: pll3_r_ck output is enabled

Bit 5 **DIVQEN**: PLL3 DIVQ divider output enable

Set and reset by software to enable the pll3_q_ck output of the PLL3.

In order to save power, when the pll3_q_ck is not needed, DIVQEN and DIVQ must be set to '0'.

0: pll3_q_ck output is disabled (default after reset)
1: pll3_q_ck output is enabled

Bit 4 **DIVPEN**: PLL3 DIVP divider output enable

Set and reset by software to enable the pll3_p_ck output of the PLL3.

In order to save power, when the pll3_p_ck is not needed, DIVPEN and DIVP must be set to '0'.

0: pll3_p_ck output is disabled (default after reset)
1: pll3_p_ck output is enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **PLL3RDY**: PLL3 clock ready flag

Set by hardware to indicate that the PLL3 is locked.

0: PLL3 unlocked (default after reset)
1: PLL3 locked

Bit 0 **PLLON**: PLL3 enable

Set and cleared by software to enable the PLL3.

Cleared by hardware when entering Stop, LP-Stop, LPLV-Stop, or Standby mode.

Note that DIVPEN, DIVQEN and DIVREN of PLL3 must be set to ‘0’ before setting PLLON to ‘0’. refer to *Section : PLL disabling procedure* for details.

0: PLL3 OFF (default after reset)
1: PLL3 ON, and ref3_ck is provided to the PLL3

10.7.39 RCC PLL3 configuration register 1 (RCC_PLL3CFGR1)

Address offset: 0x884

Reset value: 0x0001 0031

This register is used to configure the PLL3. If TZEN = MCKPROT = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	IFRGE[1:0]		Res.	Res.	DIVM3[5:0]					
						rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVN[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **IFRGE[1:0]**: PLL3 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL3.
x0: The PLL3 input (ref3_ck) clock range frequency is between 4 and 8 MHz (default after reset)
x1: The PLL3 input (ref3_ck) clock range frequency is between 8 and 16 MHz.
Note that if ref3_ck is equal to 8 MHz, it is recommended to set IFRGE = ‘x1’

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DIVM3[5:0]**: Prescaler for PLL3
Set and cleared by software to configure the prescaler of the PLL3.
0x0: bypass
0x1: division by 2 (default after reset)
0x2: division by 3
...
0x3F: division by 64

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DIVN[8:0]**: Multiplication factor for PLL3 VCO
Set and reset by software to control the multiplication factor of the VCO.
Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is between 400 and 800 MHz.

VCO output frequency = Fref3_ck x (DIVN+1), when value 0 has been loaded into FRACV, with:
- Valid division ratios for DIVN: between 25 and 200
- The input frequency Fref3_ck between 4 and 16 MHz

0x18: Division ratio is 25
0x19: Division ratio is 26
0x1A: Division ratio is 27
...
0x31: Division ratio is 50 (default after reset)
...
0xC7: Division ratio is 200
Others: wrong configurations

10.7.40 RCC PLL3 configuration register 2 (RCC_PLL3CFGR2)

Address offset: 0x888

Reset value: 0x0001 0101

This register is used to configure the PLL3. If TZEN = MCKPROT = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DIVQ[6:0]							Res.	DIVP[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **DIVR[6:0]**: PLL3 DIVR division factor

Set and reset by software to control the frequency of the pll3_r_ck clock.

0x0: pll3_r_ck = vco3_ck
0x1: pll3_r_ck = vco3_ck / 2 (default after reset)
0x2: pll3_r_ck = vco3_ck / 3
...
0x7F: pll3_r_ck = vco3_ck / 128

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **DIVQ[6:0]**: PLL3 DIVQ division factor

Set and reset by software to control the frequency of the pll3_q_ck clock.

0x0: pll3_q_ck = vco3_ck
0x1: pll3_q_ck = vco3_ck / 2 (default after reset)
0x2: pll3_q_ck = vco3_ck / 3
...
0x7F: pll3_q_ck = vco3_ck / 128

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DIVP[6:0]**: PLL3 DIVP division factor

Set and reset by software to control the frequency of the pll3_p_ck clock.

0x0: pll3_p_ck = vco3_ck
0x1: pll3_p_ck = vco3_ck / 2 (default after reset)
0x2: pll3_p_ck = vco3_ck / 3
...
0x7F: pll3_p_ck = vco3_ck / 128

10.7.41 RCC PLL3 fractional register (RCC_PLL3FRACR)

Address offset: 0x88C

Reset value: 0x0000 0000

This register is used to fine-tune the frequency of the PLL3 VCO. If TZEN = MCKPROT = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACLE
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRACV[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **FRACLE**: PLL3 fractional latch enable

Set and reset by software to latch the content of FRACV into the Sigma-Delta modulator.

In order to latch the FRACV value into the Sigma-Delta modulator, FRACLE must be set to '0', then set to '1': the transition 0 to '1' transfers the content of FRACV into the modulator. Refer to *Section : PLL initialization procedure* and *Section : Reprogramming post-dividers* for additional information on the PLL programming.

Bits 15:3 **FRACV[12:0]**: Fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is between 400 and 800 MHz.

VCO output frequency = Fref3_ck x ((DIVN+1) + (FRACV / 8192)), with

- DIVN shall be between 4 and 512
- FRACV can be between 0 and 8191
- The input frequency Fref3_ck shall be between 1 and 16 MHz

In order to change the FRACV value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit FRACLE to '0',
- write the new fractional value into FRACV,
- set the bit FRACLE to '1'.

Bits 2:0 Reserved, must be kept at reset value.

10.7.42 RCC PLL4 control register (RCC_PLL4CR)

Address offset: 0x894

Reset value: 0x0000 0000

This register is used to control the PLL4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVREN	DIVQEN	DIVPEN	Res.	Res.	PLL4RDY	PLLON
									rw	rw	rw			r	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DIVREN**: PLL4 DIVR divider output enable

Set and reset by software to enable the pll4_r_ck output of the PLL4.

In order to save power, when the pll4_r_ck is not needed, DIVREN must be set to '0'.

0: pll4_r_ck output is disabled (default after reset)
1: pll4_r_ck output is enabled

Bit 5 **DIVQEN**: PLL4 DIVQ divider output enable

Set and reset by software to enable the pll4_q_ck output of the PLL4.

In order to save power, when the pll4_q_ck is not needed, DIVQEN must be set to '0'.

0: pll4_q_ck output is disabled (default after reset)
1: pll4_q_ck output is enabled

Bit 4 **DIVPEN**: PLL4 DIVP divider output enable

Set and reset by software to enable the pll4_p_ck output of the PLL4.

In order to save power, when the pll4_p_ck is not needed, DIVPEN must be set to '0'.

0: pll4_p_ck output is disabled (default after reset)
1: pll4_p_ck output is enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **PLL4RDY**: PLL4 clock ready flag

Set by hardware to indicate that the PLL4 is locked.

0: PLL4 unlocked (default after reset)
1: PLL4 locked

Bit 0 **PLLON**: PLL4 enable

Set and cleared by software to enable the PLL4.

Cleared by hardware when entering Stop, LP-Stop, LPLV-Stop, or Standby mode.

Note that DIVPEN, DIVQEN and DIVREN of PLL4 must be set to ‘0’ before setting PLLON to ‘0’. refer to *Section : PLL disabling procedure* for details.

0: PLL4 OFF (default after reset)
1: PLL4 ON, and ref4_ck is provided to the PLL4

10.7.43 RCC PLL4 configuration register 1 (RCC_PLL4CFGR1)

Address offset: 0x898

Reset value: 0x0001 0031

This register is used to configure the PLL4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	IFRGE[1:0]		Res.	Res.	DIVM4[5:0]					
						rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVN[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **IFRGE[1:0]**: PLL4 input frequency range

Set and reset by software to select the proper reference frequency range used for PLL4.

x0: The PLL4 input (ref4_ck) clock range frequency is between 4 and 8 MHz (default after reset)

x1: The PLL4 input (ref4_ck) clock range frequency is between 8 and 16 MHz

Note that if ref3_ck is equal to 8 MHz, it is recommended to set IFRGE = ‘x1’

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DIVM4[5:0]**: Prescaler for PLL4

Set and cleared by software to configure the prescaler of the PLL4.

0x0: bypass
0x1: division by 2 (default after reset)
0x2: division by 3
...
0x3F: division by 64

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DIVN[8:0]**: Multiplication factor for PLL4 VCO

Set and reset by software to control the multiplication factor of the VCO.

Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is between 400 and 800 MHz.

VCO output frequency = Fref4_ck x (DIVN+1), when value 0 has been loaded into FRACV, with:

- Valid division ratios for DIVN: between 25 and 200
- The input frequency Fref4_ck between 4 and 16 MHz

0x18: Division ratio is 25
0x19: Division ratio is 26
0x1A: Division ratio is 27
...
0x31: Division ratio is 50 (default after reset)
...
0xC7: Division ratio is 200
Others: wrong configurations

10.7.44 RCC PLL4 configuration register 2 (RCC_PLL4CFGR2)

Address offset: 0x89C

Reset value: 0x0000 0000

This register is used to configure the PLL4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DIVQ[6:0]							Res.	DIVP[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **DIVR[6:0]**: PLL4 DIVR division factor

Set and reset by software to control the frequency of the pll4_r_ck clock.

0x0: pll4_r_ck = vco4_ck (default after reset)
0x1: pll4_r_ck = vco4_ck / 2
0x2: pll4_r_ck = vco4_ck / 3
...
0x7F: pll4_r_ck = vco4_ck / 128

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **DIVQ[6:0]**: PLL4 DIVQ division factor

Set and reset by software to control the frequency of the pll4_q_ck clock.

0x0: pll4_q_ck = vco4_ck (default after reset)
0x1: pll4_q_ck = vco4_ck / 2
0x2: pll4_q_ck = vco4_ck / 3
...
0x7F: pll4_q_ck = vco4_ck / 128

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DIVP[6:0]**: PLL4 DIVP division factor

Set and reset by software to control the frequency of the pll4_p_ck clock.

0x0: pll4_p_ck = vco4_ck (default after reset)
0x1: pll4_p_ck = vco4_ck / 2
0x2: pll4_p_ck = vco4_ck / 3
...
0x7F: pll4_p_ck = vco4_ck / 128

10.7.45 RCC PLL4 fractional register (RCC_PLL4FRACR)

Address offset: 0x8A0

Reset value: 0x0000 0000

This register is used to fine-tune the frequency of the PLL4 VCO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACLE
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRACV[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **FRACLE**: PLL4 fractional latch enable

Set and reset by software to latch the content of FRACV into the Sigma-Delta modulator.

In order to latch the FRACV value into the Sigma-Delta modulator, FRACLE must be set to '0', then set to '1': the transition 0 to '1' transfers the content of FRACV into the modulator. Refer to Section: Refer to *Section : PLL initialization procedure* and *Section : Reprogramming post-dividers* for additional information.

Bits 15:3 **FRACV[12:0]**: Fractional part of the multiplication factor for PLL4 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL4 VCO.

Warning: the software has to set correctly these bits to ensure that the VCO output frequency is between its valid frequency range, which is between 400 and 800 MHz.

VCO output frequency = Fref4_ck x ((DIVN+1) + (FRACV / 8192)), with

- DIVN shall be between 4 and 512
- FRACV can be between 0 and 8191
- The input frequency Fref4_ck shall be between 1 and 16 MHz

In order to change the FRACV value on-the-fly even if the PLL is enabled, the application has to proceed as follow:

- set the bit FRACLE to '0',
- write the new fractional value into FRACV,
- set the bit FRACLE to '1'.

Bits 2:0 Reserved, must be kept at reset value.

10.7.46 RCC I2C1,2 kernel clock selection register (RCC_I2C12CKSELR)

Address offset: 0x8C0

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the I2C1 and I2C2. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C12SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **I2C12SRC[2:0]**: I2C1 and I2C2 kernel clock source selection

Set and reset by software.

0x0: pclk1 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_r_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.47 RCC I2C3,5 kernel clock selection register (RCC_I2C35CKSELR)

Address offset: 0x8C4

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the I2C3 and I2C5. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C35SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **I2C35SRC[2:0]**: I2C3 and I2C5 kernel clock source selection

Set and reset by software.

0x0: pclk1 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_r_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.48 RCC I2C4, 6 kernel clock selection register (RCC_I2C46CKSELR)

Address offset: 0x0C0

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the I2C4 and I2C6. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C46SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **I2C46SRC[2:0]**: I2C4 and I2C6 kernel clock source selection

Set and reset by software.

In secure mode, only the mux positions 0 and 1 guarantee that the kernel clock will not be modified by non-secure accesses.

0x0: pclk5 clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.49 RCC SAI1 kernel clock selection register (RCC_SAI1CKSELR)

Address offset: 0x8C8

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SAI1 and DFSDM audio clock. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI1SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SAI1SRC[2:0]**: SAI1 and DFSDM kernel clock source selection

Set and reset by software.

0x0: pll4_q_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: I2S_CKIN clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock
0x4: pll3_r_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.50 RCC SAI2 kernel clock selection register (RCC_SAI2CKSELR)

Address offset: 0x8CC

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SAI2. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SAI2SRC[2:0]**: SAI2 kernel clock source selection

Set and reset by software.

0x0: pll4_q_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: I2S_CKIN clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock
0x4: spdif_ck_symb clock from SPDIFRX selected as kernel peripheral clock
0x5: pll3_r_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.51 RCC SAI3 kernel clock selection register (RCC_SAI3CKSELR)

Address offset: 0x8D0

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SAI3. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI3SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SAI3SRC[2:0]**: SAI3 kernel clock source selection
Set and reset by software.
0x0: pll4_q_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: I2S_CKIN clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock
0x4: pll3_r_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.52 RCC SAI4 kernel clock selection register (RCC_SAI4CKSELR)

Address offset: 0x8D4

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SAI4. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SAI4SRC[2:0]**: SAI4 kernel clock source selection
Set and reset by software.
0x0: pll4_q_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: I2S_CKIN clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock
0x4: pll3_r_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.53 RCC SPI/I2S1 kernel clock selection register (RCC_SPI2S1CKSELR)

Address offset: 0x8D8

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SPI/I2S1. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI1SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SPI1SRC[2:0]**: SPI/I2S1 kernel clock source selection
Set and reset by software.

0x0: pll4_p_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: I2S_CKIN clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock
0x4: pll3_r_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.54 RCC SPI/I2S2,3 kernel clock selection register (RCC_SPI2S23CKSELR)

Address offset: 0x8DC

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SPI/I2S2,3. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI23SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SPI23SRC[2:0]**: SPI/I2S2,3 kernel clock source selection

Set and reset by software.

0x0: pll4_p_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: I2S_CKIN clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock
0x4: pll3_r_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.55 RCC SPI4,5 kernel clock selection register (RCC_SPI45CKSELR)

Address offset: 0x8E0

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SPI4,5. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI45SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SPI45SRC[2:0]**: SPI4,5 kernel clock source selection

Set and reset by software.

0x0: pclk2 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: hse_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.56 RCC SPI6 kernel clock selection register (RCC_SPI6CKSELR)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SPI6. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI6SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SPI6SRC[2:0]**: SPI6 kernel clock source selection

Set and reset by software.

In secure mode, only the mux positions 0 and 1 guarantee that the kernel clock will not be modified by non-secure accesses.

0x0: pclk5 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: hse_ker_ck clock selected as kernel peripheral clock
0x5: pll3_q_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.57 RCC USART6 kernel clock selection register (RCC_UART6CKSELR)

Address offset: 0x8E4

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the USART6. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART6SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **UART6SRC[2:0]**: USART6 kernel clock source selection

Set and reset by software.

0x0: pclk2 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: hse_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.58 RCC UART2,4 kernel clock selection register (RCC_UART24CKSELR)

Address offset: 0x8E8

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the USART2 and UART4. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART24SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **UART24SRC[2:0]**: USART2 and UART4 kernel clock source selection

Set and reset by software.

0x0: pclk1 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: hse_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.59 RCC UART3,5 kernel clock selection register (RCC_UART35CKSELR)

Address offset: 0x8EC

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the USART3 and UART5. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART35SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **UART35SRC[2:0]**: USART3 and UART5 kernel clock source selection
Set and reset by software.

0x0: pclk1 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: hse_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.60 RCC UART7,8 kernel clock selection register (RCC_UART78CKSELR)

Address offset: 0x8F0

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the UART7 and UART8. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART78SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **UART78SRC[2:0]**: UART7 and UART8 kernel clock source selection

Set and reset by software.

0x0: pclk1 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: hse_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.61 RCC USART1 kernel clock selection register (RCC_UART1CKSELR)

Address offset: 0x0C8

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the USART1. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART1SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **UART1SRC[2:0]**: UART1 kernel clock source selection
Set and reset by software.
0x0: pclk5 clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
0x3: csi_ker_ck clock selected as kernel peripheral clock
0x4: pll4_q_ck clock selected as kernel peripheral clock
0x5: hse_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.62 RCC SDMMC1 and 2 kernel clock selection register (RCC_SDMMC12CKSELR)

Address offset: 0x8F4

Reset value: 0x0000 0003

This register is used to control the selection of the kernel clock for the SDMMC1 and SDMMC2. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC12SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SDMMC12SRC[2:0]**: SDMMC1 and SDMMC2 kernel clock source selection

Set and reset by software.

0x0: hclk6 clock selected as kernel peripheral clock
0x1: pll3_r_ck clock selected as kernel peripheral clock
0x2: pll4_p_ck clock selected as kernel peripheral clock
0x3: hsi_ker_ck clock selected as kernel peripheral clock (default after reset)
others: reserved, the kernel clock is disabled

10.7.63 RCC SDMMC3 kernel clock selection register (RCC_SDMMC3CKSELR)

Address offset: 0x8F8

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SDMMC3. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **SDMMC3SRC[2:0]**: SDMMC3 kernel clock source selection

Set and reset by software.

0x0: hclk2 clock selected as kernel peripheral clock (default after reset)
0x1: pll3_r_ck clock selected as kernel peripheral clock
0x2: pll4_p_ck clock selected as kernel peripheral clock
0x3: hsi_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.64 RCC ethernet kernel clock selection register (RCC_ETHCKSELR)

Address offset: 0x8FC

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the ETH block. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETHPTPDIV[3:0]				Res.	Res.	ETHSRC[1:0]	
								rw	rw	rw	rw			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **ETHPTPDIV[3:0]**: Clock divider for Ethernet Precision Time Protocol (PTP)

Set and cleared by software to configure the divider value.

0x0: bypass (default after reset)
0x1: division by 2
0x2: division by 3
...
0xF: division by 16

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **ETHSRC[1:0]**: ETH kernel clock source selection

Set and reset by software.

0x0: pll4_p_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
others: the kernel clock is disabled

10.7.65 RCC QUADSPI kernel clock selection register (RCC_QSPICKSELR)

Address offset: 0x900

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the QUADSPI. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **QSPISRC[1:0]**: QUADSPI kernel clock source selection
Set and reset by software.

0x0: aclk clock selected as kernel peripheral clock (default after reset)
0x1: pll3_r_ck clock selected as kernel peripheral clock
0x2: pll4_p_ck clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock

10.7.66 RCC FMC kernel clock selection register (RCC_FMCCKSELR)

Address offset: 0x904

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the FMC block. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **FMCSRC[1:0]**: FMC kernel clock source selection

Set and reset by software.

0x0: aclk clock selected as kernel peripheral clock (default after reset)
0x1: pll3_r_ck clock selected as kernel peripheral clock
0x2: pll4_p_ck clock selected as kernel peripheral clock
0x3: per_ck clock selected as kernel peripheral clock

10.7.67 RCC FDCAN kernel clock selection register (RCC_FDCANCKSELR)

Address offset: 0x90C

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the FDCAN block. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **FDCANSRC[1:0]**: FDCAN kernel clock source selection

Set and reset by software.

0x0: hse_ker_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: pll4_q_ck clock selected as kernel peripheral clock
0x3: pll4_r_ck clock selected as kernel peripheral clock

10.7.68 RCC SPDIFRX kernel clock selection register (RCC_SPDIFCKSELR)

Address offset: 0x914

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the SPDIFRX. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPDIFSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SPDIFSRC[1:0]**: SPDIFRX kernel clock source selection
Set and reset by software.

0x0: pll4_p_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll3_q_ck clock selected as kernel peripheral clock
0x2: hsi_ker_ck clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.69 RCC CEC kernel clock selection register (RCC_CECCKSELR)

Address offset: 0x918

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the CEC-HDMI. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CECSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **CECSRC[1:0]**: CEC-HDMI kernel clock source selection
Set and reset by software.
0x0: lse_ck clock selected as kernel peripheral clock (default after reset)
0x1: lsi_ck clock selected as kernel peripheral clock
0x2: csi_ker_ck/122 clock selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

10.7.70 RCC USB kernel clock selection register (RCC_USBCKSELR)

Address offset: 0x91C

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the USBPHY PLL of the USB HOST and USB OTG. It also controls the pre-divider for the reference clock for the USBPHY. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOSRC	Res.	Res.	USBPHYSRC[1:0]	
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **USBOSRC**: USB OTG kernel clock source selection
Set and reset by software.
0: pll4_r_ck clock selected as kernel peripheral clock (default after reset)
1: clock provided by the USB PHY (rcc_ck_usbo_48m) selected as kernel peripheral clock

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **USBPHYSRC[1:0]**: USB PHY kernel clock source selection
Set and reset by software.
0x0: hse_ker_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll4_r_ck clock selected as kernel peripheral clock
0x2: hse_ker_ck/2 clock selected as kernel peripheral clock
other: Clock disabled

10.7.71 RCC RNG1 kernel clock selection register (RCC_RNG1CKSELR)

Address offset: 0x0CC

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the RNG1. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1SRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **RNG1SRC[1:0]**: RNG1 kernel clock source selection

Set and reset by software.

0x0: csi_ker_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll4_r_ck clock selected as kernel peripheral clock
0x2: lse_ck clock selected as kernel peripheral clock
0x3: lsi_ck clock selected as kernel peripheral clock

10.7.72 RCC RNG2 kernel clock selection register (RCC_RNG2CKSELR)

Address offset: 0x920

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the RNG2. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG2SRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **RNG2SRC[1:0]**: RNG2 kernel clock source selection

Set and reset by software.

0x0: csi_ker_ck clock selected as kernel peripheral clock (default after reset)
0x1: pll4_r_ck clock selected as kernel peripheral clock
0x2: lse_ck clock selected as kernel peripheral clock
0x3: lsi_ck clock selected as kernel peripheral clock

10.7.73 RCC common peripheral clock selection register (RCC_CPERCKSELR)

Address offset: 0x0D0

Reset value: 0x0000 0000

This register is used to select an oscillator source as kernel clock for the per_ck clock. The per_ck clock is distributed to several peripherals. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKPERSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **CKPERSRC[1:0]**: Oscillator selection for kernel clock

Set and cleared by software.

0x0: hsi_ker_ck clock selected (default after reset)
0x1: csi_ker_ck clock selected
0x2: hse_ker_ck clock selected
0x3: Clock disabled

10.7.74 RCC STGEN clock selection register (RCC_STGENCKSELR)

Address offset: 0x0D4

Reset value: 0x0000 0000

This register is used to select the peripheral clock for the STGEN block. Note that this clock is used to provide a time reference for the application. Refer to *Section : Clock enabling delays*. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **STGENSRC[1:0]**: Oscillator selection for kernel clock

Set and cleared by software.

0x0: hsi_ker_ck clock selected (default after reset)
0x1: hse_ker_ck clock selected
others: Clock disabled

10.7.75 RCC DDR interface control register (RCC_DDRITFCR)

Address offset: 0x0D8

Reset value: 0x000F D02A

This register is used to control the DDR interface, including the DDRC and DDRPHYC. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GSKP_DUR[3:0]				DFILP_WIDTH[2:0]			GSKPCTRL	GSKPMOD	DDRCKMOD[2:0]			DPHYCTLRST	DPHYRST	DPHYAPBRST	DDRCORERST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DDRCAXIRST	DDRCAPBRST	KERDCG_DLY[2:0]			DDRPHYCAPBLPEN	DDRPHYCAPBEN	AXIDCGEN	DDRCAPBLPEN	DDRCAPBEN	DDRPHYCLPEN	DDRPHYCEN	DDRC2LPEN	DDRC2EN	DDRC1LPEN	DDRC1EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **GSKP_DUR[3:0]**: Glitch skipper duration in automatic mode
Set and clear by software.
This field allow the application to define the amount of time where the glitch skipper is providing the clock dphy_ker_ck instead of the clock ddrc_ker_ck to the DDRC block.
This delay is equal to (GSKP_DUR + 1) x 32 x Tdphy_ker_ck
0x0: Sets a delay of 32 x Tdphy_ker_ck
0x1: Sets a delay of 2 x 32 x Tdphy_ker_ck
0x2: Sets a delay of 3 x 32 x Tdphy_ker_ck
...
0xF: Sets a delay of 16 x 32 x Tdphy_ker_ck

Bits 27:25 **DFILP_WIDTH[2:0]**: Minimum duration of low-power request command
Set and clear by software.
The low power request duration must always be bigger or equal to 1 microsecond. If this duration is not respected, the DDR interface behavior is not guaranteed. The application has to program properly DFILP_WIDTH in order to ensure a delay bigger or equal to 1 microsecond. Note that the delay is also dependent of the frequency of the dphy_ker_ck clock.
0x0: Bypass, delay disabled
0x1: Forces a delay of 160 x Tdphy_ker_ck to be used when Fdphy_ker_ck is between 120 and 160 MHz.
0x2: Forces a delay of 224 x Tdphy_ker_ck to be used when Fdphy_ker_ck is between 160 and 220 MHz.
0x3: Forces a delay of 320 x Tdphy_ker_ck to be used when Fdphy_ker_ck is between 220 and 320 MHz.
0x4: Forces a delay of 416 x Tdphy_ker_ck to be used when Fdphy_ker_ck is between 320 and 410 MHz.
Others: Forces a delay of 544 x Tdphy_ker_ck to be used when Fdphy_ker_ck is between 410 and 533MHz.

Bit 24 **GSKPCTRL**: Glitch Skipper (GSKP) control
Set and clear by software.
This has effect only if GSKPMOD = '0'.
0: The GSKP block is providing the clock phy_out_ck (provided by the DDRPHYC)
1: The GSKP block is providing the clock dphy_ker_ck (provided by the RCC)

Bit 23 **GSKPMOD**: Glitch Skipper (GSKP) Mode
Set and clear by software.
0: The GSKP block is controlled by the GSKPCTRL bit.
1: The GSKP block is controlled automatically by the DFI.

Bits 22:20 **DDRCKMOD[2:0]**: RCC mode for DDR clock control

Set and clear by software.

0x0: Normal mode: the gating of the dphy_ker_ck clock depends on the DDRPHYCEN, DDRPHYCLPEN and MPU mode. The gating of the ddrc_ker_ckg clock depends on the DDRCxEN, DDRCxLPEN and MPU mode. This mode must be selected during DDRC and DDRPHYC initialization phase, and if the application is using the software self-refresh (SSR).

0x1: Automatic Self-Refresh mode (ASR1): the clock ddrc_ker_ckg is gated automatically according to cactive_ddrc signal. The gating of the dphy_ker_ck clock depends on the DDRPHYCEN, DDRPHYCLPEN and MPU mode.
This mode can be used when the application wants to maintain a clock to the DLLs in order to avoid the re-lock sequences. In this mode it is recommended to have DDRCxEN = DDRCxLPEN = DDRPHYCEN = DDRPHYCLPEN = '1'.

0x5: Full Automatic Self-Refresh mode (ASR2): the clocks ddrc_ker_ckg and dphy_ker_ck are gated automatically according to cactive_ddrc signal.
This mode can be used when the DLLs are in bypass. In this mode it is recommended to have DDRCxEN = DDRCxLPEN = DDRPHYCEN = DDRPHYCLPEN = '1'.

0x2: Hardware Self-Refresh mode (HSR1): the gating of the ddrc_ker_ckg clock is controlled by the AXI-Low-Power interface connected to the DDRC. The gating of the dphy_ker_ck clock depends on the DDRPHYCEN, DDRPHYCLPEN and MPU mode.
This mode can be used when the application wants to maintain a clock to the DLLs. In this mode it is recommended to have DDRCxEN = DDRCxLPEN = DDRPHYCEN = DDRPHYCLPEN = '1'.

0x6: Full Hardware Self-Refresh mode (HSR2): the gating of ddrc_ker_ckg and dphy_ker_ck clocks are controlled by the AXI-Low-Power interface connected to the DDRC.
This mode can be used when the DLLs are in bypass. In this mode it is recommended to have DDRCxEN = DDRCxLPEN = DDRPHYCEN = DDRPHYCLPEN = '1'.

other: reserved

Bit 19 **DPHYCTLRST**: DDRPHYC Control reset

Set and clear by software.

0: does not reset the DDRPHYC Control
1: resets the DDRPHYC Control

Bit 18 **DPHYRST**: DDRPHYC reset

Set and clear by software.

0: does not reset the DDRPHYC
1: resets the DDRPHYC

Bit 17 **DPHYAPBRST**: DDRPHYC APB interface reset

Set and clear by software.

0: does not reset the DDRPHYC APB interface
1: resets the DDRPHYC APB interface

Bit 16 **DDRCORERST**: DDRC core reset

Set and clear by software.

0: does not reset the DDRC core
1: resets the DDRC core

Bit 15 **DDRCAXIRST**: DDRC AXI interface reset
Set and clear by software.
0: does not reset the DDRC AXI interface
1: resets the DDRC AXI interface

Bit 14 **DDRCAPBRST**: DDRC APB interface reset
Set and clear by software.
0: does not reset the DDRC APB interface
1: resets the DDRC APB interface

Bits 13:11 **KERDCG_DLY[2:0]**: AXIDCG delay
Set and clear by software.
It represents the delay between the falling edge of the cactive_ddrc signal and the moment where the KERDCG will gate the ddrc_ker_ckg.
0x0: 1 period of ddrc_ker_ck between cactive_ddrc falling edge and the gating of ddrc_ker_ckg.
0x1: 3 periods of ddrc_ker_ck between cactive_ddrc falling edge and the gating of ddrc_ker_ckg.
...
0x7: 15 periods of ddrc_ker_ck between cactive_ddrc falling edge and the gating of ddrc_ker_ckg.

Bit 10 **DDRPHYCAPBLPEN**: DDRPHYC APB clock enable during CSleep mode
Set and clear by software.
This bit is controlling the gating of pclk4_dphy clock during CSleep mode.
0: means that the APB clock is disabled in CSleep
1: means that the APB clock is enabled in CSleep

Bit 9 **DDRPHYCAPBEN**: DDRPHYC APB clock enable
Set and clear by software.
This bit is controlling the gating of pclk4_dphy clock.
0: means that the APB clock is disabled
1: means that the APB clock is enabled

Bit 8 **AXIDCGEN**: AXIDCG enable during MPU CRun mode
Set and clear by software.
When the dynamic clock gating is disabled, the clock is always provided to the DDRC if provided by the RCC.
0: means that the dynamic clock gating of AXIDCG[2:1] is disabled during MPU CRun,
1: means that the dynamic clock gating of AXIDCG{2:1] is enabled during MPU CRun

Bit 7 **DDRCAPBLPEN**: DDRC APB clock enable during CSleep mode
Set and clear by software.
This bit is controlling the gating of pclk4_ddrc clock during CSleep mode.
0: means that the APB clock is disabled in CSleep
1: means that the APB clock is enabled in CSleep

Bit 6 **DDRCAPBEN**: DDRC APB clock enable
Set and clear by software.
This bit is controlling the gating of pclk4_ddrc clock.
0: means that the APB clock is disabled
1: means that the APB clock is enabled

Bit 5 **DDRPHYCLPEN**: DDRPHYC peripheral clocks enable during CSleep mode
Set and clear by software.
This bit is controlling the gating of dphy_ker_ck and pclk4_dphy in CSleep
0: means that the peripheral clocks are disabled in CSleep
1: means that the peripheral clocks are enabled in CSleep

Bit 4 **DDRPHYCEN**: DDRPHYC peripheral clocks enable
Set and clear by software.
This bit is controlling the gating of dphy_ker_ck and pclk4_dphy
0: means that the peripheral clocks are disabled
1: means that the peripheral clocks are enabled

Bit 3 **DDRC2LPEN**: DDRC port 2 peripheral clocks enable during CSleep mode
Set and clear by software.
This bit is controlling the gating of ddrc_ker_ckg, aclk2_ddrc and pclk4_ddrc in CSleep.
0: means that the peripheral clocks are disabled in CSleep
1: means that the peripheral clocks are enabled in CSleep

Bit 2 **DDRC2EN**: DDRC port 2 peripheral clocks enable
Set and clear by software.
This bit is controlling the gating of ddrc_ker_ckg, aclk2_ddrc and pclk4_ddrc clocks.
0: Means that the DDRC peripheral clocks are disabled
1: Means that the DDRC peripheral clocks are enabled

Bit 1 **DDRC1LPEN**: DDRC port 1 peripheral clocks enable during CSleep mode
Set and clear by software.
This bit is controlling the gating of ddrc_ker_ckg, aclk1_ddrc and pclk4_ddrc in CSleep.
0: means that the peripheral clocks are disabled in CSleep
1: means that the peripheral clocks are enabled in CSleep

Bit 0 **DDRC1EN**: DDRC port 1 peripheral clocks enable
Set and clear by software.
This bit is controlling the gating of ddrc_ker_ckg, aclk1_ddrc and pclk4_ddrc clocks.
0: Means that the DDRC peripheral clocks are disabled
1: Means that the DDRC peripheral clocks are enabled

10.7.76 RCC ADC kernel clock selection register (RCC_ADCCKSELR)

Address offset: 0x928

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the ADC block. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCSRC[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **ADCSRC[1:0]**: ADC1 and 2 kernel clock source selection

Set and reset by software.

0x0: pll4_r_ck clock selected as kernel peripheral clock (default after reset)
0x1: per_ck clock selected as kernel peripheral clock
0x2: pll3_q_ck clock selected as kernel peripheral clock
others: the kernel clock is disabled

10.7.77 RCC LPTIM4 and 5 kernel clock selection register (RCC_LPTIM45CKSELR)

Address offset: 0x92C

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the LPTIM4 and LPTIM5 blocks. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM45SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **LPTIM45SRC[2:0]**: LPTIM4 and LPTIM5 kernel clock source selection

Set and reset by software.

0x0: pclk3 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_p_ck clock selected as kernel peripheral clock
0x2: pll3_q_ck clock selected as kernel peripheral clock
0x3: lse_ck clock selected as kernel peripheral clock
0x4: lsi_ck clock selected as kernel peripheral clock
0x5: per_ck clock selected as kernel peripheral clock
others: the kernel clock is disabled

10.7.78 RCC LPTIM2 and 3 kernel clock selection register (RCC_LPTIM23CKSELR)

Address offset: 0x930

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the LPTIM2 and LPTIM3 blocks. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM23SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **LPTIM23SRC[2:0]**: LPTIM2 and LPTIM3 kernel clock source selection

Set and reset by software.

0x0: pclk3 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_q_ck clock selected as kernel peripheral clock
0x2: per_ck clock selected as kernel peripheral clock
0x3: lse_ck clock selected as kernel peripheral clock
0x4: lsi_ck clock selected as kernel peripheral clock
others: the kernel clock is disabled

10.7.79 RCC LPTIM1 kernel clock selection register (RCC_LPTIM1CKSELR)

Address offset: 0x934

Reset value: 0x0000 0000

This register is used to control the selection of the kernel clock for the LPTIM1 block. Note that changing the clock source on-the-fly is allowed, and will not generate any timing violation, however the user has to ensure that both the previous and the new clock sources are present during the switching, and for the whole transition time. Refer to *Section : Clock enabling delays*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SRC[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **LPTIM1SRC[2:0]**: LPTIM1 kernel clock source selection

Set and reset by software.

0x0: pclk1 clock selected as kernel peripheral clock (default after reset)
0x1: pll4_p_ck clock selected as kernel peripheral clock
0x2: pll3_q_ck clock selected as kernel peripheral clock
0x3: lse_ck clock selected as kernel peripheral clock
0x4: lsi_ck clock selected as kernel peripheral clock
0x5: per_ck clock selected as kernel peripheral clock
others: the kernel clock is disabled

10.7.80 RCC boot control register (RCC_MP_BOOTCR)

Address offset: 0x100

Reset value: 0x0000 0000

This register is used to control the HOLD boot function when the system exits from Standby. Refer to *Section : MCU HOLD_BOOT after processor reset*. This register is reset when a system reset occurs, but not when the circuit exits from Standby (app_rst reset).

If TZEN = '1', this register can only be modified in secure mode. This register can only be accessed by the MPU.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPU_BEN	MCU_BEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MPU_BEN**: MPU Boot Enable after Standby

Set and reset by software in order to allow or not the MPU to restart the next time the system exits from Standby. The value of this bit is used by the BOOTROM in order to allow or not the restart of the MPU application.

0: The MPU will remain on CStandby when the system exits from Standby

1: The MPU is allowed to restart when the system exits from Standby

Bit 0 **MCU_BEN**: MCU Boot Enable after Standby

Set and reset by software in order to allow or not the MCU to restart the next time the system exits from Standby. The value of this bit is used by the BOOTROM in order to allow or not the restart of the MPU application.

0: The MCU will remain on HOLD_BOOT when the system exits from Standby

1: The MCU is allowed to restart when the system exits from Standby

10.7.81 RCC stop request set register (RCC_MP_SREQSETR)

Address offset: 0x104

Reset value: 0x0000 0000

Writing '0' has no effect, reading will return the values of the bits. Writing a '1' sets the corresponding bit to '1'. The MCU cannot write to this register. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P1	STPREQ_P0
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STPREQ_P1**: Stop Request for MPU processor number 1
Set by software
Indicates if the MPU processor number 1 allows the entry in CStop mode for MPU domain, when the complete MPU is in WFI.
0: Writing '0' has no effect, reading '0' means that the MPU processor number 1 does not allow the MPU domain to go to CStop
1: Writing '1' sets the STPREQ_P1 bit, reading '1' means that the MPU processor number 1 allows the MPU domain to go to CStop

Bit 0 **STPREQ_P0**: Stop Request for MPU processor number 0
Set by software
Indicates if the MPU processor number 0 allows the entry in CStop mode for MPU domain, when the complete MPU is in WFI.
0: Writing '0' has no effect, reading '0' means that the MPU processor number 0 does not allow the MPU domain to go to CStop
1: Writing '1' sets the STPREQ_P0 bit, reading '1' means that the MPU processor number 0 allows the MPU domain to go to CStop

10.7.82 RCC stop request clear register (RCC_MP_SREQCLRR)

Address offset: 0x108

Reset value: 0x0000 0000

Writing '0' has no effect, reading will return the effective values of the bits. Writing a '1' sets the corresponding bit to '0'. The MCU cannot write to this register. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P1	STPREQ_P0
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STPREQ_P1**: Stop Request for MPU processor number 1

Cleared by software

Indicates if the MPU processor number 1 allows the entry in CStop mode for MPU domain, when the complete MPU is in WFI.

0: Writing '0' has no effect, reading '0' means that the MPU processor number 1 does not allow the MPU domain to go to CStop

1: Writing '1' clears the STPREQ_P1 bit, reading '1' means that the MPU processor number 1 allows the MPU domain to go to CStop

Bit 0 **STPREQ_P0**: Stop Request for MPU processor number 0

Cleared by software

Indicates if the MPU processor number 0 allows the entry in CStop mode for MPU domain, when the complete MPU is in WFI.

0: Writing '0' has no effect, reading '0' means that the MPU processor number 0 does not allow the MPU domain to go to CStop

1: Writing '1' clears the STPREQ_P0 bit, reading '1' means that the MPU processor number 0 allows the MPU domain to go to CStop

10.7.83 RCC global control register (RCC_MP_GCR)

Address offset: 0x10C

Reset value: 0x0000 0000

The register contains global control bits. If TZEN = '1', this register can only be modified in secure mode. The MCU cannot modify this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_MCU
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **BOOT_MCU**: Allows the MCU to boot

Set and reset by software in order set the MCU in HOLD_BOOT after a reset of the MCU core. Note that the MCU core can be reset by mpu_rst or by nreset.

Refer to *Section : MCU HOLD_BOOT after processor reset* for details.

If the MCU was in HOLD_BOOT it will cause the MCU to boot.

0: The MCU will be set in HOLD_BOOT when the next MCU core reset occurs. (default after reset)

1: The MCU will not be in HOLD_BOOT mode when the next MCU core reset occurs.

10.7.84 RCC application reset control register (RCC_MP_APRSTCR)

Address offset: 0x110

Reset value: 0x0000 7F00

This register is used to control the behavior of the warm reset. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RSTTO[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDCTLEN
	rw	rw	rw	rw	rw	rw	rw								rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **RSTTO[6:0]**: Reset Timeout Delay Adjust

Set and reset by software in order to define the maximum amount of time the app_rst is delayed. The timeout value is counted using hsi_ck, which can be divided according to HSIDIV.

This function is only available when the RDCTLEN = '1'.

It is not allowed to change this field value when RDCTLEN = ‘1’. Refer to *Section 10.3.11: Reset delay control (RDCTL)* for additional information of the programming of the timeout value.

0x0: The timeout function is disabled
0x1: The timeout is set to 2 x 2^{HSIDIV} us
...
0x7F: The timeout is set to 128 x 2^{HSIDIV} us (default after reset)

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RDCTLEN**: Reset Delay Control Enable

Set and reset by software.

0: The RDCTL control block is bypassed (default after reset)
1: The RDCTL control block is enabled.

10.7.85 RCC application reset status register (RCC_MP_APRSTSR)

Address offset: 0x114

Reset value: 0x0000 0000

This register provides a status of the RDCTL. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RSTTOV[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r								

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **RSTTOV[6:0]**: Reset Timeout Delay Value

Set by hardware in order to give the value of the timeout counter. This function is only available when the RDCTLEN = '1'.

If RSTTOS = '0' it means that a timeout occurred.

Bits 7:0 Reserved, must be kept at reset value.

10.7.86 RCC backup domain control register (RCC_BDCR)

Address offset: 0x140

Reset value: 0x0000 0020

This register is used to control the LSE function. Wait states are inserted in case of successive write accesses to this register. The number of wait states may be up to 7 cycles of AHB4 clock.

After a system reset, the register RCC_BDCR is write-protected. In order to modify this register, the DBP bit in the PWR control register 1 (PWR_CR1) has to be set to '1'. Bits of RCC_BDCR register are only reset after a backup domain reset: nreset_vsw (see *Section 10.3.6: Backup domain reset*). Any other internal or external reset will not have any effect on these bits.

This register is located into the V_{SW} domain. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VSWRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCCKEN	Res.	Res.	RTCSRC[1:0]	
rw											rw			rwo	rwo

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	LSECSSD	LSECSSON	Res.	Res.	LSEDRV[1:0]		DIGBYP	LSERDY	LSEBYP	LSEON
						r	rw			rw	rw	rw	r	rw	rw

Bit 31 **VSWRST**: V Switch domain software reset

Set and reset by software.

0: Reset not activated (default after backup domain reset)

1: Resets the entire V_{SW} domain

Bits 30:21 Reserved, must be kept at reset value.

Bit 20 **RTCCKEN**: RTC clock enable

Set and reset by software.

0: rtc_ck clock is disabled (default after backup domain reset)

1: rtc_ck clock enabled

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **RTCSRC[1:0]**: RTC clock source selection

Set by software to select the clock source for the RTC. This field can be written only one time after backup domain reset.

When a LSE failure occurs, the RTCSRC is set to '0', and the software is allowed to write it again once.

This field must be written before LSECSSON is enabled.

Refer to *Section : CSS on LSE* for details.

0x0: No clock (default after backup domain reset)

0x1: LSE clock used as RTC clock

0x2: LSI clock used as RTC clock

0x3: HSE clock divided by RTCDIV value is used as RTC clock

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSD**: LSE clock security system failure detection

Set by hardware to indicate when a failure has been detected by the Clock Security System on the external 32 kHz oscillator.

0: No failure detected on 32 kHz oscillator (default after backup domain reset)
1: Failure detected on 32 kHz oscillator

Bit 8 **LSECSSON**: LSE clock security system enable

Set by software to enable the Clock Security System on 32 kHz oscillator.

Refer to *Section : CSS on LSE* for details on the activation and deactivation sequences.

Once the LSECSSON bit is enabled it cannot be disabled, except after a LSE failure detection (LSECSSD = '1').

0: Clock Security System on 32 kHz oscillator OFF (default after backup domain reset)
1: Clock Security System on 32 kHz oscillator ON

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **LSEDRV[1:0]**: LSE oscillator driving capability

Set by software to select the driving capability of the LSE oscillator.

0x0: Lowest drive
0x1: Medium low drive
0x2: Medium high drive (default after backup domain reset)
0x3: Highest drive

Bit 3 **DIGBYP**: LSE digital bypass

Set and reset by software to select the analog or digital bypass mode.

0: The LSE is in analog bypass mode (default after backup domain reset)

1: The LSE is in digital bypass mode

Bit 2 **LSERDY**: LSE oscillator ready

Set and reset by hardware to indicate when the LSE is stable. This bit needs 6 cycles of lse_ck clock to fall down after LSEON has been set to '0'.

0: LSE oscillator not ready (default after backup domain reset)
1: LSE oscillator ready

Bit 1 **LSEBYP**: LSE oscillator bypass

Set and reset by software to bypass oscillator in debug mode.

This bit must not be written when the LSE is enabled (LSEON = '1').

Refer to *Section : Note as well that the signal csi_cal_ck is gated when the system goes to Stop.* for details.

0: LSE oscillator not bypassed (default after backup domain reset)
1: LSE oscillator bypassed

Bit 0 **LSEON**: LSE oscillator enabled

Set and reset by software.

0: LSE oscillator OFF (default after backup domain reset)
1: LSE oscillator ON

10.7.87 RCC reset duration and LSI control register (RCC_RDLSICR)

Address offset: 0x144

Reset value: 0x0000 0000

This register is used to control the minimum NRST active duration and LSI function.

0 to 7 wait states are inserted for word, half-word and byte accesses. Wait states are inserted in case of successive accesses to this register.

This register is reset by the por_rst reset, and it is located into the V_{DD} domain. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPARE[4:0]					EADLY[2:0]			Res.	Res.	Res.	MRD[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
														r	rw

Bits 31:27 **SPARE[4:0]**: Spare bits

Set and reset by software. Reserved for future use.

Bits 26:24 **EADLY[2:0]**: External access delays

Set and reset by software.

Time to wait before the BOOTROM performs any external access (UART, USB, QUADSPI, FMC, SDMMC,...)

0x0: 10 ms (default after reset)
0x1: No extra delay added by the BOOTROM
0x2: 100 us
0x3: 200 us
0x4: 500 us
0x5: 1 ms
0x6: 2 ms
0x7: 5 ms

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **MRD[4:0]**: Minimum Reset Duration

Set and reset by software. They define the minimum guaranteed duration of the NRST low pulse. The LSI oscillator is automatically enabled when needed by the RPCTL.

0x0: NRST low pulse duration is guaranteed by the pulse stretcher of the PAD. The RPCTL is bypassed (default after reset)
0x1: The guaranteed NRST low pulse duration is about 1 ms (1 x 32 lsi_ck cycles),
0x2: The guaranteed NRST low pulse duration is about 2 ms (2 x 32 lsi_ck cycles),
...
0x1F: The guaranteed NRST low pulse duration is about 31 ms (31 x 32 lsi_ck cycles).

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LSIRDY**: LSI oscillator ready

Set and reset by hardware to indicate when the LSI is stable. This bit needs 3 cycles of lsi_ck clock to fall down after LSION has been set to '0'.

0: LSI oscillator not ready (default after reset)
1: LSI oscillator ready

Bit 0 **LSION**: LSI oscillator enabled

Set and reset by software.

This bit is used to request the activation of the LSI oscillator. Note that IWDG1 and IWDG2 can also request the activation of LSI.

0: LSI oscillator is OFF if no other requester needs it, such as IWDG[2:1] (default after reset)
1: LSI oscillator ON

10.7.88 RCC clock source interrupt enable register (RCC_MP_CIER)

Address offset: 0x414

Reset value: 0x0000 0000

This register shall be used by the MPU to control the interrupt source enable. Refer to *Section 10.5: RCC interrupts* for more details. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE
											rw				rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	PLL4DYIE	PLL3DYIE	PLL2DYIE	PLL1DYIE	Res.	Res.	Res.	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPIE**: Wake up from CStop Interrupt Enable

Set and reset by software to enable/disable the generation of the interrupt used to wake up the MPU from CStop.

0: Wakeup interrupt disabled (default after reset)

1: Wakeup interrupt enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSIE**: LSE clock security system Interrupt Enable

Set and reset by software to enable/disable interrupt caused by the Clock Security System on external 32 kHz oscillator.

0: LSE CSS interrupt disabled (default after reset)

1: LSE CSS interrupt enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **PLL4DYIE**: PLL4 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL4 lock.

0: PLL4 lock interrupt disabled (default after reset)

1: PLL4 lock interrupt enabled

Bit 10 **PLL3DYIE**: PLL3 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL3 lock.

0: PLL3 lock interrupt disabled (default after reset)

1: PLL3 lock interrupt enabled

Bit 9 **PLL2DYIE**: PLL2 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL2 lock.

0: PLL2 lock interrupt disabled (default after reset)

1: PLL2 lock interrupt enabled

Bit 8 **PLL1DYIE**: PLL1 ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CSIRDYIE**: CSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.
0: CSI ready interrupt disabled (default after reset)
1: CSI ready interrupt enabled

Bit 3 **HSERDYIE**: HSE ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 2 **HSIRDYIE**: HSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

10.7.89 RCC clock source interrupt flag register (RCC_MP_CIFR)

Address offset: 0x418

Reset value: 0x0000 0000

This register shall be used by the MPU in order to read and clear the interrupt flags.Writing '0' has no effect, writing '1' will clear the corresponding flag.Refer to *Section 10.5: RCC interrupts* for more details. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF
											rc_w1				rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	PLL4DYF	PLL3DYF	PLL2DYF	PLL1DYF	Res.	Res.	Res.	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPF**: Wake up from CStop Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the MPU needs to exit from CStop.

0: No wakeup interrupt pending (default after reset)

1: Wakeup interrupt pending, writing '1' clears this flag

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSF**: LSE clock security system Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when a failure is detected on the external 32 kHz oscillator.

0: No failure detected on the external 32 kHz oscillator (default after reset)

1: A failure is detected on the external 32 kHz oscillator, writing '1' clears this flag

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **PLL4DYF**: PLL4 ready Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the PLL4 locks.

0: No clock ready interrupt caused by PLL4 lock (default after reset)

1: Clock ready interrupt caused by PLL4 lock, writing '1' clears this flag

Bit 10 **PLL3DYF**: PLL3 ready Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the PLL3 locks.

0: No clock ready interrupt caused by PLL3 lock (default after reset)

1: Clock ready interrupt caused by PLL3 lock, writing '1' clears this flag

Bit 9 **PLL2DYF**: PLL2 ready Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the PLL2 locks.

0: No clock ready interrupt caused by PLL2 lock (default after reset)

1: Clock ready interrupt caused by PLL2 lock, writing '1' clears this flag

Bit 8 **PLL1DYF**: PLL1 ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the PLL1 locks.
0: No clock ready interrupt caused by PLL1 lock (default after reset)
1: Clock ready interrupt caused by PLL1 lock, writing '1' clears this flag

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CSIRDYF**: CSI ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the CSI clock becomes stable.
0: No clock ready interrupt caused by the CSI (default after reset)
1: Clock ready interrupt caused by the CSI, writing '1' clears this flag

Bit 3 **HSERDYF**: HSE ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the HSE clock becomes stable.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE, writing '1' clears this flag

Bit 2 **HSIRDYF**: HSI ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the HSI clock becomes stable.
0: No clock ready interrupt caused by the HSI (default after reset)
1: Clock ready interrupt caused by the HSI, writing '1' clears this flag

Bit 1 **LSERDYF**: LSE ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the LSE clock becomes stable.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE, writing '1' clears this flag

Bit 0 **LSIRDYF**: LSI ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the LSI clock becomes stable.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI, writing '1' clears this flag

10.7.90 RCC clock source interrupt enable register (RCC_MC_CIER)

Address offset: 0xC14

Reset value: 0x0000 0000

This register shall be used by the MCU to control the interrupt source enable. Refer to *Section 10.5: RCC interrupts* for more details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE
											rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4DYIE	PLL3DYIE	PLL2DYIE	PLL1DYIE	Res.	Res.	Res.	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPIE**: Wake up from CStop Interrupt Enable

Set and reset by software to enable/disable the generation of the interrupt used to wake up the MCU from CStop.

0: Wak-up interrupt disabled (default after reset)

1: Wakeup interrupt enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSIE**: LSE clock security system Interrupt Enable

Set and reset by software to enable/disable interrupt caused by the Clock Security System on external 32 kHz oscillator.

Note that the LSE security system feature must also be enabled (LSECSSON = '1'), in order to generate interrupts.

0: LSE CSS interrupt disabled (default after reset)

1: LSE CSS interrupt enabled

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **PLL4DYIE**: PLL4 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL4 lock.

0: PLL4 lock interrupt disabled (default after reset)

1: PLL4 lock interrupt enabled

Bit 10 **PLL3DYIE**: PLL3 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL3 lock.

0: PLL3 lock interrupt disabled (default after reset)

1: PLL3 lock interrupt enabled

Bit 9 **PLL2DYIE**: PLL2 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL2 lock.

0: PLL2 lock interrupt disabled (default after reset)

1: PLL2 lock interrupt enabled

Bit 8 **PLL1DYIE**: PLL1 ready Interrupt Enable

Set and reset by software to enable/disable interrupt caused by PLL1 lock.

0: PLL1 lock interrupt disabled (default after reset)

1: PLL1 lock interrupt enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CSIRDYIE**: CSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.
0: CSI ready interrupt disabled (default after reset)
1: CSI ready interrupt enabled

Bit 3 **HSERDYIE**: HSE ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 2 **HSIRDYIE**: HSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready Interrupt Enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

10.7.91 RCC clock source interrupt flag register (RCC_MC_CIFR)

Address offset: 0xC18

Reset value: 0x0000 0000

This register shall be used by the MCU in order to read and clear the interrupt flags.Writing '0' has no effect, writing '1' will clear the corresponding flag.Refer to *Section 10.5: RCC interrupts* for more details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF
											rc_w1				rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	PLL4DYF	PLL3DYF	PLL2DYF	PLL1DYF	Res.	Res.	Res.	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WKUPF**: Wake up from CStop Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the MCU needs to exit from CStop.

0: No wakeup interrupt pending (default after reset)

1: Wakeup interrupt pending, writing '1' clears this flag

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **LSECSSF**: LSE clock security system Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when a failure is detected on the external 32 kHz oscillator.

0: No failure detected on the external 32 kHz oscillator (default after reset)

1: A failure is detected on the external 32 kHz oscillator, writing '1' clears this flag

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **PLL4DYF**: PLL4 ready Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the PLL4 locks.

0: No clock ready interrupt caused by PLL4 lock (default after reset)

1: Clock ready interrupt caused by PLL4 lock, writing '1' clears this flag

Bit 10 **PLL3DYF**: PLL3 ready Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the PLL3 locks.

0: No clock ready interrupt caused by PLL3 lock (default after reset)

1: Clock ready interrupt caused by PLL3 lock, writing '1' clears this flag

Bit 9 **PLL2DYF**: PLL2 ready Interrupt Flag

Reset by software by writing this flag to '1'.

Set by hardware when the PLL2 locks.

0: No clock ready interrupt caused by PLL2 lock (default after reset)

1: Clock ready interrupt caused by PLL2 lock, writing '1' clears this flag

Bit 8 **PLL1DYF**: PLL1 ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the PLL1 locks.
0: No clock ready interrupt caused by PLL1 lock (default after reset)
1: Clock ready interrupt caused by PLL1 lock, writing '1' clears this flag

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CSIRDYF**: CSI ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the CSI clock becomes stable.
0: No clock ready interrupt caused by the CSI (default after reset)
1: Clock ready interrupt caused by the CSI, writing '1' clears this flag

Bit 3 **HSERDYF**: HSE ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the HSE clock becomes stable.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE, writing '1' clears this flag

Bit 2 **HSIRDYF**: HSI ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the HSI clock becomes stable.
0: No clock ready interrupt caused by the HSI (default after reset)
1: Clock ready interrupt caused by the HSI, writing '1' clears this flag

Bit 1 **LSERDYF**: LSE ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the LSE clock becomes stable.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE, writing '1' clears this flag

Bit 0 **LSIRDYF**: LSI ready Interrupt Flag
Reset by software by writing this flag to '1'.
Set by hardware when the LSI clock becomes stable.
0: No clock ready interrupt caused by the LSI (default after reset)
1: Clock ready interrupt caused by the LSI, writing '1' clears this flag

10.7.92 RCC PWR_LP delay control register (RCC_PWRLPDLYCR)

Address offset: 0x41C

Reset value: 0x0000 0000

This register is used to program the delay between the moment where the system exits from one of the Stop modes, and the moment where it is allowed to enable the PLLs and provide a clock to bridges and processors. If TZEN = '1', this register can only be modified in secure mode. Write access to this register is not allowed during the clock restore sequence. See *Section : The clock restore sequence description* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCTMPSKP	Res.	Res.	PWRLP_DLY[21:16]					
							rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWRLP_DLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MCTMPSKP**: Skip the PWR_DLY for MCU
Set and Reset by software.
This bit is used by the clock restore sequence to skip the PWRLP_TEMPO Delay for the restore of the mcuss_ck when the system exits from one of the Stop modes.
0: The clock restore sequence of the MCU waits for the PWRLP_TEMPO delay before activating power consuming elements (default after reset)
1: The clock restore sequence of the MCU runs the PWRLP_TEMPO delay but did not wait for the delay to elapse before activating power consuming elements

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:0 **PWRLP_DLY[21:0]**: PWRLP_TEMPO value
Set and Reset by software.
This register contains the delay value to be observed before activating the PLLs and interconnect clocks, after one of the system Stop.
The delay is counted with the HSI clock. The user has to take into account HSIDIV value.
The delay value is given by the following formula: PWRLP_DLY[21:0] / FHSI
0x0: The PWRLP_TEMPO delay is bypassed (default after reset)
0x1: A PWRLP_TEMPO delay of one period of HSI (at 64 MHz) is observed
...
0x3FFFFF: A PWRLP_TEMPO delay of about 65.5 milliseconds is observed

10.7.93 RCC APB1 peripheral reset set register (RCC_APB1RSTSETR)

Address offset: 0x980

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSRST	Res.	DAC12RST	Res.	CECRST	SPDIFRST	Res.	I2C5RST	I2C3RST	I2C2RST	I2C1RST	Res.	UART8RST	UART7RST	UART5RST	UART4RST
rs		rs		rs	rs		rs	rs	rs	rs		rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rs	rs		rs	rs		rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bit 31 **MDIOSRST**: MDIOS block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12RST**: DAC1 and 2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECRST**: HDMI-CEC block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 26 **SPDIFRST**: SPDIFRX block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5RST**: I2C5 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 23 **I2C3RST**: I2C3 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 22 **I2C2RST**: I2C2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 21 **I2C1RST**: I2C1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8RST**: UART8 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 18 **UART7RST**: UART7 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 17 **UART5RST**: UART5 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 16 **UART4RST**: UART4 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 15 **USART3RST**: USART3 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 14 **USART2RST**: USART2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3RST**: SPI3 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 11 **SPI2RST**: SPI2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 8 **TIM14RST**: TIM14 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 7 **TIM13RST**: TIM13 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 6 **TIM12RST**: TIM12 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 5 **TIM7RST**: TIM7 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 4 **TIM6RST**: TIM6 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 3 **TIM5RST**: TIM5 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 2 **TIM4RST**: TIM4 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 1 **TIM3RST**: TIM3 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 0 **TIM2RST**: TIM2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.94 RCC APB1 peripheral reset clear register (RCC_APB1RSTCLRR)

Address offset: 0x984

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSRST	Res.	DAC12RST	Res.	CECRST	SPDIFRST	Res.	I2C5RST	I2C3RST	I2C2RST	I2C1RST	Res.	UART8RST	UART7RST	UART5RST	UART4RST
rc_w1		rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **MDIOSRST**: MDIOS block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12RST**: DAC1 and 2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECRST**: HDMI-CEC block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 26 **SPDIFRST**: SPDIFRX block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5RST**: I2C5 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 23 **I2C3RST**: I2C3 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 22 **I2C2RST**: I2C2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 21 **I2C1RST**: I2C1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8RST**: UART8 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 18 **UART7RST**: UART7 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 17 **UART5RST**: UART5 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 16 **UART4RST**: UART4 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 15 **USART3RST**: USART3 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 14 **USART2RST**: USART2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3RST**: SPI3 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 11 **SPI2RST**: SPI2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 8 **TIM14RST**: TIM14 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 7 **TIM13RST**: TIM13 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 6 **TIM12RST**: TIM12 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 5 **TIM7RST**: TIM7 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 4 **TIM6RST**: TIM6 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 3 **TIM5RST**: TIM5 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 2 **TIM4RST**: TIM4 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 1 **TIM3RST**: TIM3 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

Bit 0 **TIM2RST**: TIM2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' release the block reset, reading '1' means that the block reset is asserted

10.7.95 RCC APB2 peripheral reset set register (RCC_APB2RSTSETR)

Address offset: 0x988

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	DFSDMRST	Res.	SAI3RST	SAI2RST	SAI1RST
							rs				rs		rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	USART6RST	Res.	Res.	SPI5RST	SPI4RST	SPI1RST	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	TIM8RST	TIM1RST
		rs			rs	rs	rs				rs	rs	rs	rs	rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANRST**: FDCAN block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **DFSDMRST**: DFSDM block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3RST**: SAI3 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 17 **SAI2RST**: SAI2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 16 **SAI1RST**: SAI1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6RST**: USART6 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5RST**: SPI5 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 9 **SPI4RST**: SPI4 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 8 **SPI1RST**: SPI/I2S1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17RST**: TIM17 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 3 **TIM16RST**: TIM16 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 2 **TIM15RST**: TIM15 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 1 **TIM8RST**: TIM8 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 0 **TIM1RST**: TIM1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.96 RCC APB2 peripheral reset clear register (RCC_APB2RSTCLRR)

Address offset: 0x98C

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	DFSDMRST	Res.	SAI3RST	SAI2RST	SAI1RST
							rc_w1				rc_w1		rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	USART6RST	Res.	Res.	SPI5RST	SPI4RST	SPI1RST	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	TIM8RST	TIM1RST
		rc_w1			rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANRST**: FDCAN block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **DFSDMRST**: DFSDM block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3RST**: SAI3 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 17 **SAI2RST**: SAI2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 16 **SAI1RST**: SAI1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6RST**: USART6 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5RST**: SPI5 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 9 **SPI4RST**: SPI4 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 8 **SPI1RST**: SPI/I2S1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17RST**: TIM17 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 3 **TIM16RST**: TIM16 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 2 **TIM15RST**: TIM15 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 1 **TIM8RST**: TIM8 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 0 **TIM1RST**: TIM1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.97 RCC APB3 peripheral reset set register (RCC_APB3RSTSETR)

Address offset: 0x990

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRST
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFRST	Res.	SYSCFGRST	Res.	Res.	SAI4RST	Res.	Res.	Res.	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST
		rs		rs			rs					rs	rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DTSRST**: DTS block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFRST**: VREF block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGRST**: SYSCFG block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4RST**: SAI4 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5RST**: LPTIM5 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 2 **LPTIM4RST**: LPTIM4 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 1 **LPTIM3RST**: LPTIM3 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 0 **LPTIM2RST**: LPTIM2 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.98 RCC APB3 peripheral reset clear register (RCC_APB3RSTCLRR)

Address offset: 0x994

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRST
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFRST	Res.	SYSCFGRST	Res.	Res.	SAI4RST	Res.	Res.	Res.	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST
		rc_w1		rc_w1			rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DTSRST**: DTS block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFRST**: VREF block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGRST**: SYSCFG block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4RST**: SAI4 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5RST**: LPTIM5 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 2 **LPTIM4RST**: LPTIM4 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 1 **LPTIM3RST**: LPTIM3 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 0 **LPTIM2RST**: LPTIM2 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.99 RCC AHB2 peripheral reset set register (RCC_AHB2RSTSETR)

Address offset: 0x998

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3RST
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBORST	Res.	Res.	ADC12RST	Res.	Res.	DMAMUXRST	DMA2RST	DMA1RST
							rs			rs			rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3RST**: SDMMC3 and the SDMMC3 delay (DLYBSD3) block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBORST**: USBO block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12RST**: ADC1 and 2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXRST**: DMAMUX block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 1 **DMA2RST**: DMA2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 0 **DMA1RST**: DMA1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.100 RCC AHB2 peripheral reset clear register (RCC_AHB2RSTCLRR)

Address offset: 0x99C

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3RST
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBORST	Res.	Res.	ADC12RST	Res.	Res.	DMAMUXRST	DMA2RST	DMA1RST
							rc_w1			rc_w1			rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3RST**: SDMMC3 and the SDMMC3 delay (DLYBSD3) block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBORST**: USBO block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12RST**: ADC1 and 2 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXRST**: DMAMUX block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 1 **DMA2RST**: DMA2 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 0 **DMA1RST**: DMA1 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.101 RCC AHB3 peripheral reset set register (RCC_AHB3RSTSETR)

Address offset: 0x9A0

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IPCCRST	HSEMRST	Res.	Res.	Res.	CRC2RST	RNG2RST	HASH2RST	CRYP2RST	Res.	Res.	Res.	DCMIRST
			rs	rs				rs	rs	rs	rs				rs

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCRST**: IPCC block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 11 **HSEMRST**: HSEM block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2RST**: CRC2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 6 **RNG2RST**: RNG2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 5 **HASH2RST**: HASH2 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 4 **CRYP2RST**: CRYP2 (3DES/AES2) block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIRST**: DCMI block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.102 RCC AHB3 peripheral reset clear register (RCC_AHB3RSTCLRR)

Address offset: 0x9A4

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IPCCRST	HSEMRST	Res.	Res.	Res.	CRC2RST	RNG2RST	HASH2RST	CRYP2RST	Res.	Res.	Res.	DCMIRST
			rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCRST**: IPCC block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 11 **HSEMRST**: HSEM block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2RST**: CRC2 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 6 **RNG2RST**: RNG2 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 5 **HASH2RST**: HASH2 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 4 **CRYP2RST**: CRYP2 (3DES/AES2) block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIRST**: DCMI block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.103 RCC AHB4 peripheral reset set register (RCC_AHB4RSTSETR)

Address offset: 0x9A8

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
					rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKRST**: GPIOK block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 9 **GPIOJRST**: GPIOJ block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 8 **GPIOIRST**: GPIOI block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 7 **GPIOHRST**: GPIOH block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 6 **GPIOGRST**: GPIOG block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 5 **GPIOFRST**: GPIOF block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 4 **GPIOERST**: GPIOE block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 3 **GPIODRST**: GPIOD block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 2 **GPIOCRST**: GPIOC block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 1 **GPIOBRST**: GPIOB block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 0 **GPIOARST**: GPIOA block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.104 RCC AHB4 peripheral reset clear register (RCC_AHB4RSTCLRR)

Address offset: 0x9AC

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKRST**: GPIOK block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 9 **GPIOJRST**: GPIOJ block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 8 **GPIOIRST**: GPIOI block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 7 **GPIOHRST**: GPIOH block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 6 **GPIOGRST**: GPIOG block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 5 **GPIOFRST**: GPIOF block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 4 **GPIOERST**: GPIOE block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 3 **GPIODRST**: GPIOD block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 2 **GPIOCRST**: GPIOC block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 1 **GPIOBRST**: GPIOB block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 0 **GPIOARST**: GPIOA block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.105 RCC APB4 peripheral reset set register (RCC_APB4RSTSETR)

Address offset: 0x180

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPHYRST
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCRST
							rs								rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYRST**: USBPHYC block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMRST**: DDRPERFM block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCRST**: LTDC block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.106 RCC APB4 peripheral reset clear register (RCC_APB4RSTCLRR)

Address offset: 0x184

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPHYRST
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCRST
							rc_w1								rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYRST**: USBPHYC block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMRST**: DDRPERFM block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCRST**: LTDC block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.107 RCC APB5 peripheral reset set register (RCC_APB5RSTSETR)

Address offset: 0x188

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENRST	Res.	Res.	Res.	Res.
											rs				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1RST	I2C6RST	I2C4RST	Res.	SPI6RST
											rs	rs	rs		rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENRST**: STGEN block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 19:5 Reserved, must be kept at reset value.

Bit 4 **USART1RST**: USART1 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 3 **I2C6RST**: I2C6 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 2 **I2C4RST**: I2C4 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6RST**: SPI6 block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.108 RCC APB5 peripheral reset clear register (RCC_APB5RSTCLRR)

Address offset: 0x18C

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENRST	Res.	Res.	Res.	Res.
											rc_w1				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1RST	I2C6RST	I2C4RST	Res.	SPI6RST
											rc_w1	rc_w1	rc_w1		rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENRST**: STGEN block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 19:5 Reserved, must be kept at reset value.

Bit 4 **USART1RST**: USART1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 3 **I2C6RST**: I2C6 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 2 **I2C4RST**: I2C4 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6RST**: SPI6 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.109 RCC AHB5 peripheral reset set register (RCC_AHB5RSTSETR)

Address offset: 0x190

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCRST
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1RST	HASH1RST	CRYP1RST	Res.	Res.	Res.	GPIOZRST
									rs	rs	rs				rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AXIMCRST**: AXIMC block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **RNG1RST**: RNG1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 5 **HASH1RST**: HASH1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 4 **CRYP1RST**: CRYP1 (3DES/AES1) block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZRST**: GPIOZ secure block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.110 RCC AHB5 peripheral reset clear register (RCC_AHB5RSTCLRR)

Address offset: 0x194

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCRST
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1RST	HASH1RST	CRYP1RST	Res.	Res.	Res.	GPIOZRST
									rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AXIMCRST**: AXIMC block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **RNG1RST**: RNG1 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 5 **HASH1RST**: HASH1 block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 4 **CRYP1RST**: CRYP1 (3DES/AES1) block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZRST**: GPIOZ secure block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.111 RCC AHB6 peripheral reset set register (RCC_AHB6RSTSETR)

Address offset: 0x198

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHRST	Res.	Res.	Res.	CRC1RST	Res.	Res.	SDMMC2RST	SDMMC1RST
							rs				rs			rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIRST	Res.	FMCRST	Res.	ETHMACRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rs		rs		rs										

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHRST**: USBH block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1RST**: CRC1 block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2RST**: SDMMC2 and the SDMMC2 delay (DLYBSD2) block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 16 **SDMMC1RST**: SDMMC1 and the SDMMC1 delay (DLYBSD1) block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIRST**: QUADSPI and the QUADSPI delay block reset
Set by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCRST**: FMC block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ETHMACRST**: ETH block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

Bits 9:0 Reserved, must be kept at reset value.

10.7.112 RCC AHB6 peripheral reset clear register (RCC_AHB6RSTCLRR)

Address offset: 0x19C

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHRST	Res.	Res.	Res.	CRC1RST	Res.	Res.	SDMMC2RST	SDMMC1RST
							rc_w1				rc_w1			rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIRST	Res.	FMCRST	Res.	ETHMACRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rc_w1		rc_w1		rc_w1										

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHRST**: USBH block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1RST**: CRC1 block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2RST**: SDMMC2 and the SDMMC2 delay (DLYBSD2) block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 16 **SDMMC1RST**: SDMMC1 and the SDMMC1 delay (DLYBSD1) block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIRST**: QUADSPI and the QUADSPI delay block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCRST**: FMC block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ETHMACRST**: ETH block reset

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

Bits 9:0 Reserved, must be kept at reset value.

10.7.113 RCC AHB6 secure peripheral reset set register (RCC_TZAHB6RSTSETR)

Address offset: 0x1A0

Reset value: 0x0000 0000

This register is used to activate the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' activates the reset of the corresponding peripheral. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMARST
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MDMARST**: MDMA block reset

Set by software.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' asserts the block reset, reading '1' means that the block reset is asserted

10.7.114 RCC AHB6 secure peripheral reset clear register (RCC_TZAHB6RSTCLRR)

Address offset: 0x1A4

Reset value: 0x0000 0000

This register is used to release the reset of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' releases the reset of the corresponding peripheral. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMARST
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MDMARST**: MDMA block reset
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the block reset is released
1: Writing '1' releases the block reset, reading '1' means that the block reset is asserted

10.7.115 RCC global reset control set register (RCC_MP_GRSTCSETR)

Address offset: 0x404

Reset value: 0x0000 0000

This register is used by the MPU in order to generate either a MCU reset or a system reset or a reset of one of the two MPU processors. Writing '0' has no effect, reading returns the effective values of the corresponding bits. Writing a '1' activates the reset. The MCU cannot modify this register. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUP1RST	MPUP0RST	Res.	Res.	MCURST	MPSYSRST
										rs	rs			rs	rs

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **MPUP1RST**: MPU processor 1 reset

Set by software, cleared by hardware when the reset command is completed.

Note that the Snoop Control Unit of the MPU (SCU) is not reset.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' generate a reset of the MPU processor 1, reading '1' means that the block reset is asserted

Bit 4 **MPUP0RST**: MPU processor 0 reset

Set by software, cleared by hardware when the reset command is completed.

Note that the Snoop Control Unit of the MPU (SCU) is not reset.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' generate a reset of the MPU processor 0, reading '1' means that the block reset is asserted

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **MCURST**: MCU reset

Set by software, cleared by hardware when the reset command is completed.

It is recommended to set the MCU to CSleep before setting the MCURST bit to ‘1’.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' generate a reset of the MCU core, reading '1' means that the block reset is asserted

Bit 0 **MPSYSRST**: System reset

Set by software, cleared by hardware.

0: Writing '0' has no effect, reading '0' means that the block reset is released

1: Writing '1' generate a system reset, see *Figure 47*.

10.7.116 RCC APB1 peripheral enable for MPU set register (RCC_MP_APB1ENSETR)

Address offset: 0xA00

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective value of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSEN	Res.	DAC12EN	Res.	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN
rs		rs		rs	rs		rs	rs	rs	rs		rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rs	rs		rs	rs		rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bit 31 **MDIOSEN**: MDIOS peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12EN**: DAC1 and 2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECEN**: HDMI-CEC peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 26 **SPDIFEN**: SPDIFRX peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5EN**: I2C5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 23 **I2C3EN**: I2C3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 22 **I2C2EN**: I2C2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 21 **I2C1EN**: I2C1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8EN**: UART8 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 18 **UART7EN**: UART7 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **UART5EN**: UART5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **UART4EN**: UART4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **USART3EN**: USART3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 14 **USART2EN**: USART2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3EN**: SPI3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **SPI2EN**: SPI2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **TIM14EN**: TIM14 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **TIM13EN**: TIM13 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **TIM12EN**: TIM12 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **TIM7EN**: TIM7 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **TIM6EN**: TIM6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM5EN**: TIM5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM4EN**: TIM4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM3EN**: TIM3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM2EN**: TIM2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.117 RCC APB1 peripheral enable for MCU set register (RCC_MC_APB1ENSETR)

Address offset: 0xA80

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return '0'. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSEN	Res.	DAC12EN	WWDG1EN	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN
rs		rs	rs	rs	rs		rs	rs	rs	rs		rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rs	rs		rs	rs		rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bit 31 **MDIOSEN**: MDIOS peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12EN**: DAC1 and 2 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 28 **WWDG1EN**: WWDG1 peripheral clocks enable

Set by software. This bit is reset by a system reset (nreset) or MCU reset (mcu_rst).

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 27 **CECEN**: HDMI-CEC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 26 **SPDIFEN**: SPDIFRX peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5EN**: I2C5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 23 **I2C3EN**: I2C3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 22 **I2C2EN**: I2C2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 21 **I2C1EN**: I2C1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8EN**: UART8 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 18 **UART7EN**: UART7 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **UART5EN**: UART5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **UART4EN**: UART4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **USART3EN**: USART3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 14 **USART2EN**: USART2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3EN**: SPI3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **SPI2EN**: SPI2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **TIM14EN**: TIM14 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **TIM13EN**: TIM13 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **TIM12EN**: TIM12 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **TIM7EN**: TIM7 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **TIM6EN**: TIM6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM5EN**: TIM5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM4EN**: TIM4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM3EN**: TIM3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM2EN**: TIM2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.118 RCC APB1 peripheral enable for MPU clear register (RCC_MP_APB1ENCLRR)

Address offset: 0xA04

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSEN	Res.	DAC12EN	Res.	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN
rc_w1		rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **MDIOSEN**: MDIOS peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12EN**: DAC1 and 2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECEN**: HDMI-CEC peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 26 **SPDIFEN**: SPDIFRX peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5EN**: I2C5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 23 **I2C3EN**: I2C3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 22 **I2C2EN**: I2C2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 21 **I2C1EN**: I2C1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8EN**: UART8 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 18 **UART7EN**: UART7 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **UART5EN**: UART5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **UART4EN**: UART4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **USART3EN**: USART3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 14 **USART2EN**: USART2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3EN**: SPI3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **SPI2EN**: SPI2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **TIM14EN**: TIM14 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **TIM13EN**: TIM13 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **TIM12EN**: TIM12 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **TIM7EN**: TIM7 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **TIM6EN**: TIM6 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM5EN**: TIM5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM4EN**: TIM4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM3EN**: TIM3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM2EN**: TIM2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.119 RCC APB1 peripheral enable for MCU clear register (RCC_MC_APB1ENCLRR)

Address offset: 0xA84

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return '0'. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSEN	Res.	DAC12EN	Res.	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN
rc_w1		rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **MDIOSEN**: MDIOS peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12EN**: DAC1 and 2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECEN**: HDMI-CEC peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 26 **SPDIFEN**: SPDIFRX peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5EN**: I2C5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 23 **I2C3EN**: I2C3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 22 **I2C2EN**: I2C2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 21 **I2C1EN**: I2C1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8EN**: UART8 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 18 **UART7EN**: UART7 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **UART5EN**: UART5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **UART4EN**: UART4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **USART3EN**: USART3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 14 **USART2EN**: USART2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3EN**: SPI3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **SPI2EN**: SPI2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **TIM14EN**: TIM14 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **TIM13EN**: TIM13 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **TIM12EN**: TIM12 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **TIM7EN**: TIM7 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **TIM6EN**: TIM6 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM5EN**: TIM5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM4EN**: TIM4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM3EN**: TIM3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM2EN**: TIM2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.120 RCC APB2 peripheral enable for MPU set register (RCC_MP_APB2ENSETR)

Address offset: 0xA08

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN
							rs			rs	rs		rs	rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
		rs			rs	rs	rs				rs	rs	rs	rs	rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANEN**: FDCAN and CANRAM peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMEN**: Audio DFSDM peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 **DFSDMEN**: DFSDM peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3EN**: SAI3 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **SAI2EN**: SAI2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SAI1EN**: SAI1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6EN**: USART6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5EN**: SPI5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **SPI4EN**: SPI4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **SPI1EN**: SPI/I2S1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17EN**: TIM17 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM16EN**: TIM16 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM15EN**: TIM15 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM8EN**: TIM8 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM1EN**: TIM1 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.121 RCC APB2 peripheral enable for MCU set register (RCC_MC_APB2ENSETR)

Address offset: 0xA88

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN
							rs			rs	rs		rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
		rs			rs	rs	rs				rs	rs	rs	rs	rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANEN**: FDCAN and CANRAM peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMEN**: Audio DFSDM peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 **DFSDMEN**: DFSDM peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3EN**: SAI3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **SAI2EN**: SAI2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SAI1EN**: SAI1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6EN**: USART6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5EN**: SPI5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **SPI4EN**: SPI4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **SPI1EN**: SPI/I2S1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17EN**: TIM17 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM16EN**: TIM16 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM15EN**: TIM15 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM8EN**: TIM8 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM1EN**: TIM1 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.122 RCC APB2 peripheral enable for MPU clear register (RCC_MP_APB2ENCLRR)

Address offset: 0xA0C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN
							rc_w1			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
		rc_w1			rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANEN**: FDCAN and CANRAM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMEN**: Audio DFSDM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 **DFSDMEN**: DFSDM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3EN**: SAI3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **SAI2EN**: SAI2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SAI1EN**: SAI1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6EN**: USART6 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5EN**: SPI5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **SPI4EN**: SPI4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **SPI1EN**: SPI/I2S1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17EN**: TIM17 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM16EN**: TIM16 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM15EN**: TIM15 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM8EN**: TIM8 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM1EN**: TIM1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.123 RCC APB2 peripheral enable for MCU clear register (RCC_MC_APB2ENCLRR)

Address offset: 0xA8C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN
							rc_w1			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
		rc_w1			rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANEN**: FDCAN and CANRAM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMEN**: Audio DFSDM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 20 **DFSDMEN**: DFSDM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3EN**: SAI3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 17 **SAI2EN**: SAI2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SAI1EN**: SAI1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6EN**: USART6 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5EN**: SPI5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **SPI4EN**: SPI4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **SPI1EN**: SPI/I2S1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17EN**: TIM17 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **TIM16EN**: TIM16 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **TIM15EN**: TIM15 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **TIM8EN**: TIM8 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **TIM1EN**: TIM1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.124 RCC APB3 peripheral enable for MPU set register (RCC_MP_APB3ENSETR)

Address offset: 0xA10

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN
											rs				rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
		rs		rs			rs					rs	rs	rs	rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDPEN**: HDP peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **DTSEN**: DTS peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFEN**: VREF peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGEN**: SYSCFG peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4EN**: SAI4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5EN**: LPTIM5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **LPTIM4EN**: LPTIM4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **LPTIM3EN**: LPTIM3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **LPTIM2EN**: LPTIM2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.125 RCC APB3 peripheral enable for MCU set register (RCC_MC_APB3ENSETR)

Address offset: 0xA90

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN
											rs				rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
		rs		rs			rs					rs	rs	rs	rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDPEN**: HDP peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **DTSEN**: DTS peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFEN**: VREF peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGEN**: SYSCFG peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4EN**: SAI4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5EN**: LPTIM5 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **LPTIM4EN**: LPTIM4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **LPTIM3EN**: LPTIM3 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **LPTIM2EN**: LPTIM2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.126 RCC APB3 peripheral enable for MPU clear register (RCC_MP_APB3ENCLRR)

Address offset: 0xA14

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN
											rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
		rc_w1		rc_w1			rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDPEN**: HDP peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **DTSEN**: DTS peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFEN**: VREF peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGEN**: SYSCFG peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4EN**: SAI4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5EN**: LPTIM5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **LPTIM4EN**: LPTIM4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **LPTIM3EN**: LPTIM3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **LPTIM2EN**: LPTIM2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.127 RCC APB3 peripheral enable for MCU clear register (RCC_MC_APB3ENCLRR)

Address offset: 0xA94

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN
											rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
		rc_w1		rc_w1			rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **HDPEN**: HDP peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **DTSEN**: DTS peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFEN**: VREF peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGEN**: SYSCFG peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4EN**: SAI4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5EN**: LPTIM5 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **LPTIM4EN**: LPTIM4 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **LPTIM3EN**: LPTIM3 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **LPTIM2EN**: LPTIM2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.128 RCC APB4 peripheral enable for MCU set register (RCC_MC_APB4ENSETR)

Address offset: 0x280

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN
											rs				rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
							rs								rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENROEN**: STGEN Read-Only interface peripheral clocks enable

Set by software.

The peripheral clocks of the STGEN Read-Only interface are the pclk4 and the kernel clock selected by STGENSRC.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYEN**: USBPHYC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMEN**: DDRPERFM APB clock enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled

1: Writing '1' enables the APB clock, reading '1' means that the APB clock is enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LTDC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.129 RCC APB4 peripheral enable for MPU set register (RCC_MP_APB4ENSETR)

Address offset: 0x200

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN
											rs				rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG2APBEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
rs							rs								rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENROEN**: STGEN Read-Only interface peripheral clocks enable

Set by software.

The peripheral clocks of the STGEN Read-Only interface are the pclk4 and the kernel clock selected by STGENSRC.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYEN**: USBPHYC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **IWDG2APBEN**: IWDG2 APB clock enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled

1: Writing '1' enables the APB clock, reading '1' means that the APB clock is enabled

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMEN**: DDRPERFM APB clock enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled

1: Writing '1' enables the APB clock, reading '1' means that the APB clock is enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LTDC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.130 RCC APB4 peripheral enable for MCU clear register (RCC_MC_APB4ENCLRR)

Address offset: 0x284

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN
											rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
							rc_w1								rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENROEN**: STGEN Read-Only interface peripheral clocks disable

Cleared by software.

The peripheral clocks of the STGEN Read-Only interface are the pclk4 and the kernel clock selected by STGENSRC.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYEN**: USBPHYC peripheral clocks disable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMEN**: DDRPERFM APB clock enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the APB clock is disabled
1: Writing '1' disables the APB clock, reading '1' means that the APB clock is enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LTDC peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.131 RCC APB4 peripheral enable for MPU clear register (RCC_MP_APB4ENCLRR)

Address offset: 0x204

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN
											rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG2APBEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
rc_w1							rc_w1								rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENROEN**: STGEN Read-Only interface peripheral clocks disable
Cleared by software.
The peripheral clocks of the STGEN Read-Only interface are the pclk4 and the kernel clock selected by STGENSRC.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYEN**: USBPHYC peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **IWDG2APBEN**: IWDG2 APB clock disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the APB clock is disabled
1: Writing '1' disables the APB clock, reading '1' means that the APB clock is enabled

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMEN**: DDRPERFM APB clock enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled
1: Writing '1' disables the APB clock, reading '1' means that the APB clock is enabled

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LTDC peripheral clocks disable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.132 RCC APB5 peripheral enable for MCU set register (RCC_MC_APB5ENSETR)

Address offset: 0x288

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN
											rs				rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
		rs	rs	rs			rs				rs	rs	rs		rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENEN**: STGEN Controller part, peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECEN**: BSEC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **TZPCEN**: TZPC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 **TZC2EN**: TZC AXI port 2 clocks enable

Set by software.

See *Section : Clock distribution for TZC* for details.

0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2

1: Writing '1' enables the pclk5 and aclk_tzc2 clocks, reading '1' means that the clocks are enabled for AXI port 2

Bit 11 **TZC1EN**: TZC AXI port 1 clocks enable
Set by software.
See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1
1: Writing '1' enables the pclk5 and aclk_tzc1 clocks, reading '1' means that the clocks are enabled for AXI port 1

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBEN**: RTC APB clock enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1EN**: USART1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **I2C6EN**: I2C6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **I2C4EN**: I2C4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6EN**: SPI6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.133 RCC APB5 peripheral enable for MPU set register (RCC_MP_APB5ENSETR)

Address offset: 0x208

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN
											rs				rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG1APBEN	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
rs		rs	rs	rs			rs				rs	rs	rs		rs

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENEN**: STGEN Controller part, peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECEN**: BSEC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **IWDG1APBEN**: IWDG1 APB clock enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled

1: Writing '1' enables the APB clock, reading '1' means that the APB clock is enabled

Bit 14 Reserved, must be kept at reset value.

Bit 13 **TZPCEN**: TZPC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 **TZC2EN**: TZC AXI port 2 clocks enable
Set by software.
See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2
1: Writing '1' enables the pclk5 and aclk_tzc2 clocks, reading '1' means that the clocks are enabled for AXI port 2

Bit 11 **TZC1EN**: TZC AXI port 1 clocks enable
Set by software.
See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1
1: Writing '1' enables the pclk5 and aclk_tzc1 clocks, reading '1' means that the clocks are enabled for AXI port 1

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBEN**: RTC APB clock enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1EN**: USART1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **I2C6EN**: I2C6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **I2C4EN**: I2C4 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6EN**: SPI6 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.134 RCC APB5 peripheral enable for MCU clear register (RCC_MC_APB5ENCLRR)

Address offset: 0x28C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN
											rc_w1				rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
		rc_w1	rc_w1	rc_w1			rc_w1				rc_w1	rc_w1	rc_w1		rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENEN**: STGEN Controller part, peripheral clocks disable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECEN**: BSEC peripheral clocks disable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **TZPCEN**: TZPC peripheral clocks disable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 **TZC2EN**: TZC AXI port 2 clocks disable

Cleared by software. See *Section : Clock distribution for TZC* for details.

0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2

1: Writing '1' disables the AXI port 2 peripheral clocks, reading '1' means that the peripheral clocks are enabled for AXI port 2

Bit 11 **TZC1EN**: TZC AXI port 1 clocks disable

Cleared by software. See *Section : Clock distribution for TZC* for details.

0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1

1: Writing '1' disables the AXI port 1 peripheral clocks, reading '1' means that the peripheral clocks are enabled for AXI port 1

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBEN**: RTC APB clock disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1EN**: USART1 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **I2C6EN**: I2C6 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **I2C4EN**: I2C4 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6EN**: SPI6 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.135 RCC APB5 peripheral enable for MPU clear register (RCC_MP_APB5ENCLRR)

Address offset: 0x20C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN
											rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG1APBEN	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
rc_w1		rc_w1	rc_w1	rc_w1			rc_w1				rc_w1	rc_w1	rc_w1		rc_w1

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **STGENEN**: STGEN Controller part, peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECEN**: BSEC peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 **IWDG1APBEN**: IWDG1 APB clock disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the APB clock is disabled
1: Writing '1' disables the APB clock, reading '1' means that the APB clock is enabled

Bit 14 Reserved, must be kept at reset value.

Bit 13 **TZPCEN**: TZPC peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 12 **TZC2EN**: TZC AXI port 2 clocks disable
Cleared by software. See *Section : Clock distribution for TZC* clocks for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2
1: Writing '1' disables the AXI port 2 peripheral clocks, reading '1' means that the peripheral clocks are enabled for AXI port 2

Bit 11 **TZC1EN**: TZC AXI port 1 clocks disable
Cleared by software. See *Section : Clock distribution for TZC* clocks for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1
1: Writing '1' disables the AXI port 1 peripheral clocks, reading '1' means that the peripheral clocks are enabled for AXI port 1

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBEN**: RTC APB clock disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1EN**: USART1 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **I2C6EN**: I2C6 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **I2C4EN**: I2C4 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6EN**: SPI6 peripheral clocks disable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.136 RCC AHB5 peripheral enable for MCU set register (RCC_MC_AHB5ENSETR)

Address offset: 0x290

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
							rs		rs	rs	rs				rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMEN**: BKPSRAM clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1EN**: RNG1 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH1EN**: HASH1 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP1EN**: CRYP1 (3DES/AES1) peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZEN**: GPIOZ Secure peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.137 RCC AHB5 peripheral enable for MPU set register (RCC_MP_AHB5ENSETR)

Address offset: 0x210

Reset value: 0x0001 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCEN
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
							rs		rs	rs	rs				rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AXIMCEN**: AXIMC clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMEN**: BKPSRAM clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1EN**: RNG1 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH1EN**: HASH1 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP1EN**: CRYP1 (3DES/AES1) peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZEN**: GPIOZ Secure peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.138 RCC AHB5 peripheral enable for MCU clear register (RCC_MC_AHB5ENCLRR)

Address offset: 0x294

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
							rc_w1		rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMEN**: BKPSRAM clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1EN**: RNG1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH1EN**: HASH1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP1EN**: CRYP1 (3DES/AES1) peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZEN**: GPIOZ Secure peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.139 RCC AHB5 peripheral enable for MPU clear register (RCC_MP_AHB5ENCLRR)

Address offset: 0x214

Reset value: 0x0001 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCEN
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
							rc_w1		rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **AXIMCEN**: AXIMC clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMEN**: BKPSRAM clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1EN**: RNG1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH1EN**: HASH1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP1EN**: CRYP1 (3DES/AES1) peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZEN**: GPIOZ Secure peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.140 RCC AHB6 peripheral enable for MPU set register (RCC_MP_AHB6ENSETR)

Address offset: 0x218

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN
							rs				rs			rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	rs		rs		rs	rs	rs	rs							rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHEN**: USBH peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1EN**: CRC1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2EN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SDMMC1EN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIEN**: QUADSPI peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCEN**: FMC peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ETHMACEN**: Ethernet MAC bus interface Clock Enable (hclk6 and aclk)
Set by software.
0: Writing '0' has no effect, reading '0' means that the bus interface clock is disabled
1: Writing '1' enables the bus interface clock, reading '1' means that the bus interface clock is enabled

Bit 9 **ETHRXEN**: Ethernet Reception Clock Enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the reception clock is disabled
1: Writing '1' enables the reception clock, reading '1' means that the reception clock is enabled

Bit 8 **ETHTXEN**: Ethernet Transmission Clock Enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled
1: Writing '1' enables the transmission clock, reading '1' means that the transmission clock is enabled

Bit 7 **ETHCKEN**: Enable of the Ethernet clock generated by the RCC (eth_ker_ck)
Set by software.
0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled
1: Writing '1' enables the eth_ker_ck clock, reading '1' means that the eth_ker_ck clock is enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.141 RCC AHB6 peripheral enable for MCU set register (RCC_MC_AHB6ENSETR)

Address offset: 0x298

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN
							rs				rs			rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	rs		rs		rs	rs	rs	rs							rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHEN**: USBH peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1EN**: CRC1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2EN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SDMMC1EN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIEN**: QUADSPI peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCEN**: FMC peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ETHMACEN**: Ethernet MAC bus interface Clock Enable (hclk6 and aclk)
Set by software.
0: Writing '0' has no effect, reading '0' means that the bus interface clock is disabled
1: Writing '1' enables the bus interface clock, reading '1' means that the bus interface clock is enabled

Bit 9 **ETHRXEN**: Ethernet Reception Clock Enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the reception clock is disabled
1: Writing '1' enables the reception clock, reading '1' means that the reception clock is enabled

Bit 8 **ETHTXEN**: Ethernet Transmission Clock Enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled
1: Writing '1' enables the transmission clock, reading '1' means that the transmission clock is enabled

Bit 7 **ETHCKEN**: Enable of the Ethernet clock generated by the RCC (eth_ker_ck)
Set by software.
0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled
1: Writing '1' enables the eth_ker_ck clock, reading '1' means that the eth_ker_ck clock is enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.142 RCC AHB6 peripheral enable for MPU clear register (RCC_MP_AHB6ENCLRR)

Address offset: 0x21C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN
							rc_w1				rc_w1			rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1							rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHEN**: USBH peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1EN**: CRC1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2EN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SDMMC1EN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIEN**: QUADSPI and QUADSPI delay block peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCEN**: FMC peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ETHMACEN**: Disable of the bus interface clock for ETH block (hclk6 and aclk)
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the bus interface clock is disabled
1: Writing '1' disables the bus interface clock, reading '1' means that the bus interface clock is enabled

Bit 9 **ETHRXEN**: Disable of the Ethernet Reception Clock
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the reception clock is disabled
1: Writing '1' disables the reception clock, reading '1' means that the reception clock is enabled

Bit 8 **ETHTXEN**: Disable of the Ethernet Transmission Clock
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled
1: Writing '1' disables the transmission clock, reading '1' means that the transmission clock is enabled

Bit 7 **ETHCKEN**: Disable of the Ethernet clock generated by the RCC (eth_ker_ck)
Cleared by software.
0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled
1: Writing '1' disables the eth_ker_ck clock, reading '1' means that the eth_ker_ck clock is enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.143 RCC AHB6 peripheral enable for MCU clear register (RCC_MC_AHB6ENCLRR)

Address offset: 0x29C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN
							rc_w1				rc_w1			rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1							rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHEN**: USBH peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1EN**: CRC1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2EN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 16 **SDMMC1EN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPIEN**: QUADSPI and QUADSPI delay block peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCEN**: FMC peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ETHMACEN**: Disable of the bus interface clock for ETH block (hclk6 and aclk)
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the bus interface clock is disabled
1: Writing '1' disables the bus interface clock, reading '1' means that the bus interface clock is enabled

Bit 9 **ETHRXEN**: Disable of the Ethernet Reception Clock
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the reception clock is disabled
1: Writing '1' disables the reception clock, reading '1' means that the reception clock is enabled

Bit 8 **ETHTXEN**: Disable of the Ethernet Transmission Clock
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled
1: Writing '1' disables the transmission clock, reading '1' means that the transmission clock is enabled

Bit 7 **ETHCKEN**: Disable of the Ethernet clock generated by the RCC (eth_ker_ck)
Cleared by software.
0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled
1: Writing '1' disables the eth_ker_ck clock, reading '1' means that the eth_ker_ck clock is enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.144 RCC AHB6 secure peripheral enable for MPU set register (RCC_MP_TZAHB6ENSETR)

Address offset: 0x220

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA peripheral clocks enable

Set by software.

Note that this bit is “ORed” with the MDMAEN bits of RCC_MP_AHB6ENSETR/CLRR and RCC_MC_AHB6ENSETR/CLRR registers.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.145 RCC AHB6 secure peripheral enable for MPU clear register (RCC_MP_TZAHB6ENCLRR)

Address offset: 0x224

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MDMAEN**: MDMA peripheral clocks enable

Cleared by software.

Note that this bit is “ORed” with the MDMAEN bits of RCC_MP_AHB6ENSETR/CLRR and RCC_MC_AHB6ENSETR/CLRR registers.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.146 RCC AHB2 peripheral enable for MPU set register (RCC_MP_AHB2ENSETR)

Address offset: 0xA18

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
							rs			rs			rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3EN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOEN**: USBO peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1 and 2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXEN**: DMAMUX peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **DMA2EN**: DMA2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **DMA1EN**: DMA1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.147 RCC AHB2 peripheral enable for MCU set register (RCC_MC_AHB2ENSETR)

Address offset: 0xA98

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
							rs			rs			rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3EN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOEN**: USBO peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1 and 2 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXEN**: DMAMUX peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **DMA2EN**: DMA2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **DMA1EN**: DMA1 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.148 RCC AHB2 peripheral enable for MPU clear register (RCC_MP_AHB2ENCLRR)

Address offset: 0xA1C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN
															rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
							rc_w1			rc_w1			rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3EN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOEN**: USBO peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1 and 2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXEN**: DMAMUX peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **DMA2EN**: DMA2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **DMA1EN**: DMA1 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.149 RCC AHB2 peripheral enable for MCU clear register (RCC_MC_AHB2ENCLRR)

Address offset: 0xA9C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
							rc_w1			rc_w1			rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3EN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOEN**: USBO peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1 and 2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXEN**: DMAMUX peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **DMA2EN**: DMA2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **DMA1EN**: DMA1 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.150 RCC AHB3 peripheral enable for MPU set register (RCC_MP_AHB3ENSETR)

Address offset: 0xA20

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
			rs	rs				rs	rs	rs	rs				rs

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCEN**: IPCC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **HSEMEN**: HSEM peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2EN**: CRC2 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **RNG2EN**: RNG2 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH2EN**: HASH2 peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP2EN**: CRYP2 (3DES/AES2) peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIEN**: DCMI peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.151 RCC AHB3 peripheral enable for MCU set register (RCC_MC_AHB3ENSETR)

Address offset: 0xAA0

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
			rs	rs				rs	rs	rs	rs				rs

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCEN**: IPCC peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **HSEMEN**: HSEM peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2EN**: CRC2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **RNG2EN**: RNG2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH2EN**: HASH2 peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP2EN**: CRYP2 (3DES/AES2) peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIEN**: DCMI peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.152 RCC AHB3 peripheral enable for MPU clear register (RCC_MP_AHB3ENCLRR)

Address offset: 0xA24

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
			rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCEN**: IPCC peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **HSEMEN**: HSEM peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2EN**: CRC2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **RNG2EN**: RNG2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH2EN**: HASH2 peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP2EN**: CRYP2 (3DES/AES2) peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIEN**: DCMI peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.153 RCC AHB3 peripheral enable for MCU clear register (RCC_MC_AHB3ENCLRR)

Address offset: 0xAA4

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
			rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCEN**: IPCC peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 11 **HSEMEN**: HSEM peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2EN**: CRC2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **RNG2EN**: RNG2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **HASH2EN**: HASH2 peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **CRYP2EN**: CRYP2 (3DES/AES2) peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMIEN**: DCMI peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.154 RCC AHB4 peripheral enable for MPU set register (RCC_MP_AHB4ENSETR)

Address offset: 0xA28

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
					rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKEN**: GPIOK peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **GPIOJEN**: GPIOJ peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **GPIOIEN**: GPIOI peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **GPIOHEN**: GPIOH peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **GPIOGEN**: GPIOG peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **GPIOFEN**: GPIOF peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **GPIOEEN**: GPIOE peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **GPIODEN**: GPIOD peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **GPIOCEN**: GPIOC peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **GPIOBEN**: GPIOB peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **GPIOAEN**: GPIOA peripheral clocks enable
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.155 RCC AHB4 peripheral enable for MCU set register (RCC_MC_AHB4ENSETR)

Address offset: 0xAA8

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
					rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKEN**: GPIOK peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **GPIOJEN**: GPIOJ peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **GPIOIEN**: GPIOI peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **GPIOHEN**: GPIOH peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **GPIOGEN**: GPIOG peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **GPIOFEN**: GPIOF peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **GPIOEEN**: GPIOE peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **GPIODEN**: GPIOD peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **GPIOCEN**: GPIOC peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **GPIOBEN**: GPIOB peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **GPIOAEN**: GPIOA peripheral clocks enable

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' enables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.156 RCC AHB4 peripheral enable for MPU clear register (RCC_MP_AHB4ENCLRR)

Address offset: 0xA2C

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKEN**: GPIOK peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **GPIOJEN**: GPIOJ peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **GPIOIEN**: GPIOI peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **GPIOHEN**: GPIOH peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **GPIOGEN**: GPIOG peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **GPIOFEN**: GPIOF peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **GPIOEEN**: GPIOE peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **GPIODEN**: GPIOD peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **GPIOCEN**: GPIOC peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **GPIOBEN**: GPIOB peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **GPIOAEN**: GPIOA peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.157 RCC AHB4 peripheral enable for MCU clear register (RCC_MC_AHB4ENCLRR)

Address offset: 0xAAC

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKEN**: GPIOK peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 9 **GPIOJEN**: GPIOJ peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 8 **GPIOIEN**: GPIOI peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 7 **GPIOHEN**: GPIOH peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 6 **GPIOGEN**: GPIOG peripheral clocks enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled

1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 5 **GPIOFEN**: GPIOF peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 4 **GPIOEEN**: GPIOE peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 3 **GPIODEN**: GPIOD peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 2 **GPIOCEN**: GPIOC peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 1 **GPIOBEN**: GPIOB peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

Bit 0 **GPIOAEN**: GPIOA peripheral clocks enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled
1: Writing '1' disables the peripheral clocks, reading '1' means that the peripheral clocks are enabled

10.7.158 RCC AXI peripheral enable for MCU set register (RCC_MC_AXIMENSETR)

Address offset: 0xAB0

Reset value: 0x0000 0000

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMEN
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRAMEN**: SYSRAM peripheral clocks enable

Set by software.

When set, this bit indicates that the SYSRAM is allocated by the MCU. It causes the system to maintain the access to this memory according also to the MCU operation modes.

0: Writing '0' has no effect, reading '0' means that the memory is not allocated by the MCU

1: Writing '1' allocates the memory to the MCU, reading '1' means that the memory is allocated to the MCU.

10.7.159 RCC AXI peripheral enable for MCU clear register (RCC_MC_AXIMENCLRR)

Address offset: 0xAB4

Reset value: 0x0000 0000

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMEN
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRAMEN**: SYSRAM peripheral clocks enable

Cleared by software.

When set, this bit indicates that the SYSRAM is allocated by the MCU. It causes the system to maintain the access to this memory according also to the MCU operation modes.

0: Writing '0' has no effect, reading '0' means that the memory is not allocated by the MCU

1: Writing '1' deallocates the memory to the MCU, reading '1' means that the memory is allocated to the MCU.

10.7.160 RCC MLAHB peripheral enable for MCU set register (RCC_MC_MLAHBENSETR)

Address offset: 0xAB8

Reset value: 0x0000 0010

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
											rs				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMEN**: RETRAM peripheral clocks enable

Set by software.

When set, this bit indicates that the RETRAM is allocated by the MCU. It causes the system to maintain the access to this memory also according to the MCU operation modes.

0: Writing '0' has no effect, reading '0' means that the memory is not allocated by the MCU

1: Writing '1' allocates the memory to the MCU, reading '1' means that the memory is allocated to the MCU.

Bits 3:0 Reserved, must be kept at reset value.

10.7.161 RCC MLAHB peripheral enable for MCU clear register (RCC_MC_MLAHBENCLRR)

Address offset: 0xABC

Reset value: 0x0000 0010

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MCU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
											rc_w1				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMEN**: RETRAM peripheral clocks enable

Cleared by software.

When set, this bit indicates that the RETRAM is allocated by the MCU. It causes the system to maintain the access to this memory also according to the MCU operation modes.

0: Writing '0' has no effect, reading '0' means that the memory is not allocated by the MCU

1: Writing '1' deallocates the memory to the MCU, reading '1' means that the memory is allocated to the MCU.

Bits 3:0 Reserved, must be kept at reset value.

10.7.162 RCC MLAHB peripheral enable for MPU set register (RCC_MP_MLAHBENSETR)

Address offset: 0xA38

Reset value: 0x0000 0010

This register is used to set the peripheral clock enable bit of the corresponding peripheral to '1'. It shall be used to allocate a peripheral to the MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
											rs				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMEN**: RETRAM peripheral clocks enable

Set by software.

When set, this bit indicates that the RETRAM is allocated by the MPU. It causes the system to maintain the access to this memory also according to the MPU operation modes.

0: Writing '0' has no effect, reading '0' means that the memory is not allocated by the MPU

1: Writing '1' allocates the memory to the MPU, reading '1' means that the memory is allocated to the MPU.

Bits 3:0 Reserved, must be kept at reset value.

10.7.163 RCC MLAHB peripheral enable for MPU clear register (RCC_MP_MLAHBENCLRR)

Address offset: 0xA3C

Reset value: 0x0000 0010

This register is used to clear the peripheral clock enable bit of the corresponding peripheral. It shall be used to deallocate a peripheral from MPU. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
											rc_w1				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMEN**: RETRAM peripheral clocks enable

Cleared by software.

When set, this bit indicates that the RETRAM is allocated by the MPU. It causes the system to maintain the access to this memory also according to the MPU operation modes.

0: Writing '0' has no effect, reading '0' means that the memory is not allocated by the MPU

1: Writing '1' deallocates the memory to the MPU, reading '1' means that the memory is allocated to the MPU.

Bits 3:0 Reserved, must be kept at reset value.

10.7.164 RCC APB1 sleep clock enable for MPU set register (RCC_MP_APB1LPENSETR)

Address offset: 0xB00

Reset value: 0xADEF DBFF

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSLPEN	Res.	DAC12LPEN	Res.	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN
rs		rs		rs	rs		rs	rs	rs	rs		rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
rs	rs		rs	rs		rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bit 31 **MDIOSLPEN**: MDIOS peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12LPEN**: DAC1 and 2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECLPEN**: HDMI-CEC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 26 **SPDIFLPEN**: SPDIFRX peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5LPEN**: I2C5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 23 **I2C3LPEN**: I2C3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 22 **I2C2LPEN**: I2C2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 21 **I2C1LPEN**: I2C1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8LPEN**: UART8 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 18 **UART7LPEN**: UART7 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **UART5LPEN**: UART5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **UART4LPEN**: UART4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **USART3LPEN**: USART3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 14 **USART2LPEN**: USART2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3LPEN**: SPI3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **SPI2LPEN**: SPI2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **TIM14LPEN**: TIM14 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **TIM13LPEN**: TIM13 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **TIM12LPEN**: TIM12 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **TIM7LPEN**: TIM7 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **TIM6LPEN**: TIM6 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM5LPEN**: TIM5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM4LPEN**: TIM4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM3LPEN**: TIM3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM2LPEN**: TIM2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.165 RCC APB1 sleep clock enable for MCU set register (RCC_MC_APB1LPENSETR)

Address offset: 0xB80

Reset value: 0xBDEF DBFF

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSLPEN	Res.	DAC12LPEN	WWDG1LPEN	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN
rs		rs	rs	rs	rs		rs	rs	rs	rs		rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
rs	rs		rs	rs		rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bit 31 **MDIOSLPEN**: MDIOS peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12LPEN**: DAC1 and 2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 28 **WWDG1LPEN**: WWDG1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 27 **CECLPEN**: HDMI-CEC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 26 **SPDIFLPEN**: SPDIFRX peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5LPEN**: I2C5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 23 **I2C3LPEN**: I2C3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 22 **I2C2LPEN**: I2C2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 21 **I2C1LPEN**: I2C1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8LPEN**: UART8 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 18 **UART7LPEN**: UART7 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **UART5LPEN**: UART5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **UART4LPEN**: UART4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **USART3LPEN**: USART3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 14 **USART2LPEN**: USART2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3LPEN**: SPI3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **SPI2LPEN**: SPI2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **TIM14LPEN**: TIM14 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **TIM13LPEN**: TIM13 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **TIM12LPEN**: TIM12 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **TIM7LPEN**: TIM7 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **TIM6LPEN**: TIM6 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM5LPEN**: TIM5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM4LPEN**: TIM4 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM3LPEN**: TIM3 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM2LPEN**: TIM2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.166 RCC APB1 sleep clock enable MPU clear register (RCC_MP_APB1LPENCLRR)

Address offset: 0xB04

Reset value: 0xADEF DBFF

This register is used by the MPU in order to clear the PERxLPEN bits of the corresponding peripherals located into the APB1 bus. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSLPEN	Res.	DAC12LPEN	Res.	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN
rc_w1		rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **MDIOSLPEN**: MDIOS peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12LPEN**: DAC1 and 2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 28 Reserved, must be kept at reset value.

Bit 27 **CECLPEN**: HDMI-CEC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 26 **SPDIFLPEN**: SPDIFRX peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5LPEN**: I2C5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 23 **I2C3LPEN**: I2C3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 22 **I2C2LPEN**: I2C2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 21 **I2C1LPEN**: I2C1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8LPEN**: UART8 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 18 **UART7LPEN**: UART7 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **UART5LPEN**: UART5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **UART4LPEN**: UART4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **USART3LPEN**: USART3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 14 **USART2LPEN**: USART2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3LPEN**: SPI3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **SPI2LPEN**: SPI2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **TIM14LPEN**: TIM14 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **TIM13LPEN**: TIM13 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **TIM12LPEN**: TIM12 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **TIM7LPEN**: TIM7 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **TIM6LPEN**: TIM6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM5LPEN**: TIM5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM4LPEN**: TIM4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM3LPEN**: TIM3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM2LPEN**: TIM2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.167 RCC APB1 sleep clock enable MCU clear register (RCC_MC_APB1LPENCLRR)

Address offset: 0xB84

Reset value: 0xBDEF DBFF

This register is used by the MCU in order to clear the PERxLPEN bits of the corresponding peripherals located into the APB1 bus. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '0'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDIOSLPEN	Res.	DAC12LPEN	WWDG1LPEN	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN
rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bit 31 **MDIOSLPEN**: MDIOS peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 30 Reserved, must be kept at reset value.

Bit 29 **DAC12LPEN**: DAC1 and 2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 28 **WWDG1LPEN**: WWDG1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 27 **CECLPEN**: HDMI-CEC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 26 **SPDIFLPEN**: SPDIFRX peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 25 Reserved, must be kept at reset value.

Bit 24 **I2C5LPEN**: I2C5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 23 **I2C3LPEN**: I2C3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 22 **I2C2LPEN**: I2C2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 21 **I2C1LPEN**: I2C1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 Reserved, must be kept at reset value.

Bit 19 **UART8LPEN**: UART8 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 18 **UART7LPEN**: UART7 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **UART5LPEN**: UART5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **UART4LPEN**: UART4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **USART3LPEN**: USART3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 14 **USART2LPEN**: USART2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **SPI3LPEN**: SPI3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **SPI2LPEN**: SPI2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **TIM14LPEN**: TIM14 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **TIM13LPEN**: TIM13 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **TIM12LPEN**: TIM12 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **TIM7LPEN**: TIM7 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **TIM6LPEN**: TIM6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM5LPEN**: TIM5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM4LPEN**: TIM4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM3LPEN**: TIM3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM2LPEN**: TIM2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.168 RCC APB2 sleep clock enable for MPU set register (RCC_MP_APB2LPENSETR)

Address offset: 0xB08

Reset value: 0x0137 271F

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN
							rs			rs	rs		rs	rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
		rs			rs	rs	rs				rs	rs	rs	rs	rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANLPEN**: FDCAN and CANRAM peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMLPEN**: Audio DFSDM peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 **DFSDMLPEN**: DFSDM peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3LPEN**: SAI3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **SAI2LPEN**: SAI2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SAI1LPEN**: SAI1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6LPEN**: USART6 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5LPEN**: SPI5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **SPI4LPEN**: SPI4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **SPI1LPEN**: SPI/I2S1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17LPEN**: TIM17 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM16LPEN**: TIM16 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM15LPEN**: TIM15 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM8LPEN**: TIM8 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM1LPEN**: TIM1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.169 RCC APB2 sleep clock enable for MCU set register (RCC_MC_APB2LPENSETR)

Address offset: 0xB88

Reset value: 0x0137 271F

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN
							rs			rs	rs		rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
		rs			rs	rs	rs				rs	rs	rs	rs	rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANLPEN**: FDCAN and CANRAM peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMLPEN**: Audio DFSDM peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 **DFSDMLPEN**: DFSDM peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3LPEN**: SAI3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **SAI2LPEN**: SAI2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SAI1LPEN**: SAI1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6LPEN**: USART6 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5LPEN**: SPI5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **SPI4LPEN**: SPI4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **SPI1LPEN**: SPI/I2S1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17LPEN**: TIM17 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM16LPEN**: TIM16 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM15LPEN**: TIM15 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM8LPEN**: TIM8 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM1LPEN**: TIM1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.170 RCC APB2 sleep clock enable for MPU clear register (RCC_MP_APB2LPENCLRR)

Address offset: 0xB0C

Reset value: 0x0137 271F

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN
							rc_w1			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
		rc_w1			rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANLPEN**: FDCAN and CANRAM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMLPEN**: Audio DFSDM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 **DFSDMLPEN**: DFSDM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3LPEN**: SAI3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **SAI2LPEN**: SAI2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SAI1LPEN**: SAI1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6LPEN**: USART6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5LPEN**: SPI5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **SPI4LPEN**: SPI4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **SPI1LPEN**: SPI/I2S1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17LPEN**: TIM17 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM16LPEN**: TIM16 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM15LPEN**: TIM15 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM8LPEN**: TIM8 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM1LPEN**: TIM1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.171 RCC APB2 sleep clock enable for MCU clear register (RCC_MC_APB2LPENCLRR)

Address offset: 0xB8C

Reset value: 0x0137 271F

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN
							rc_w1			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
		rc_w1			rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FDCANLPEN**: FDCAN and CANRAM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ADFSDMLPEN**: Audio DFSDM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 20 **DFSDMLPEN**: DFSDM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 19 Reserved, must be kept at reset value.

Bit 18 **SAI3LPEN**: SAI3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 17 **SAI2LPEN**: SAI2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SAI1LPEN**: SAI1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **USART6LPEN**: USART6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **SPI5LPEN**: SPI5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **SPI4LPEN**: SPI4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **SPI1LPEN**: SPI/I2S1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TIM17LPEN**: TIM17 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **TIM16LPEN**: TIM16 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **TIM15LPEN**: TIM15 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **TIM8LPEN**: TIM8 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **TIM1LPEN**: TIM1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.172 RCC APB3 sleep clock enable for MPU set register (RCC_MP_APB3LPENSETR)

Address offset: 0xB10

Reset value: 0x0003 290F

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
		rs		rs			rs					rs	rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DTSLPEN**: DTS peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFLPEN**: VREF peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGLPEN**: SYSCFG peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4LPEN**: SAI4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5LPEN**: LPTIM5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **LPTIM4LPEN**: LPTIM4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **LPTIM3LPEN**: LPTIM3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **LPTIM2LPEN**: LPTIM2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.173 RCC APB3 sleep clock enable for MCU set register (RCC_MC_APB3LPENSETR)

Address offset: 0xB90

Reset value: 0x0003 290F

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
		rs		rs			rs					rs	rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DTSLPEN**: DTS peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFLPEN**: VREF peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGLPEN**: SYSCFG peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4LPEN**: SAI4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5LPEN**: LPTIM5 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **LPTIM4LPEN**: LPTIM4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **LPTIM3LPEN**: LPTIM3 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **LPTIM2LPEN**: LPTIM2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.174 RCC APB3 sleep clock enable for MPU clear register (RCC_MP_APB3LPENCLRR)

Address offset: 0xB14

Reset value: 0x0003 290F

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
		rc_w1		rc_w1			rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DTSLPEN**: DTS peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFLPEN**: VREF peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGLPEN**: SYSCFG peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4LPEN**: SAI4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5LPEN**: LPTIM5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **LPTIM4LPEN**: LPTIM4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **LPTIM3LPEN**: LPTIM3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **LPTIM2LPEN**: LPTIM2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.175 RCC APB3 sleep clock enable for MCU clear register (RCC_MC_APB3LPENCLRR)

Address offset: 0xB94

Reset value: 0x0003 290F

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN
															rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
		rc_w1		rc_w1			rc_w1					rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DTSLPEN**: DTS peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **VREFLPEN**: VREF peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 Reserved, must be kept at reset value.

Bit 11 **SYSCFGLPEN**: SYSCFG peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **SAI4LPEN**: SAI4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **LPTIM5LPEN**: LPTIM5 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **LPTIM4LPEN**: LPTIM4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **LPTIM3LPEN**: LPTIM3 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **LPTIM2LPEN**: LPTIM2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.176 RCC APB4 sleep clock enable for MCU set register (RCC_MC_APB4LPENSETR)

Address offset: 0x380

Reset value: 0x0011 0111

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN
										rs	rs				rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
							rs								rs

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENROSTPEN**: STGEN Read-Only Interface, clock enable during CStop mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop
1: Writing '1' enables the peripheral clocks in CStop, reading '1' means that the peripheral clocks are enabled in CStop

Bit 20 **STGENROLPEN**: STGEN Read-Only Interface peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYLPEN**: USBPHYC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMLPEN**: DDRPERFM APB clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep
1: Writing '1' enables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCLPEN**: LTDC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.177 RCC APB4 sleep clock enable for MPU set register (RCC_MP_APB4LPENSETR)

Address offset: 0x300

Reset value: 0x0011 8111

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN
										rs	rs				rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG2APBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
rs							rs								rs

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENROSTPEN**: STGEN Read-Only Interface, clock enable during CStop mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop

1: Writing '1' enables the peripheral clocks in CStop, reading '1' means that the peripheral clocks are enabled in CStop

Bit 20 **STGENROLPEN**: STGEN Read-Only Interface peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYLPEN**: USBPHYC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **IWDG2APBLPEN**: IWDG2 APB clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep

1: Writing '1' enables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMLPEN**: DDRPERFM APB clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep

1: Writing '1' enables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCLPEN**: LTDC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.178 RCC APB4 sleep clock enable for MCU clear register (RCC_MC_APB4LPENCLRR)

Address offset: 0x384

Reset value: 0x0011 0111

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN
										rc_w1	rc_w1				rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
							rc_w1								rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENROSTPEN**: STGEN Read-Only Interface clock enable during CStop mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop
1: Writing '1' disables the clock in CStop, reading '1' means that the clock are enabled in CStop

Bit 20 **STGENROLPEN**: STGEN Read-Only Interface peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYLPEN**: USBPHYC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMLPEN**: DDRPERFM APB clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep
1: Writing '1' disables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCLPEN**: LTDC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.179 RCC APB4 sleep clock enable for MPU clear register (RCC_MP_APB4LPENCLRR)

Address offset: 0x304

Reset value: 0x0011 8111

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN
										rc_w1	rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG2APBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
rc_w1							rc_w1								rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENROSTPEN**: STGEN Read-Only Interface clock enable during CStop mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop
1: Writing '1' disables the clock in CStop, reading '1' means that the clock are enabled in CStop

Bit 20 **STGENROLPEN**: STGEN Read-Only Interface peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **USBPHYLPEN**: USBPHYC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **IWDG2APBLPEN**: IWDG2 APB clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep
1: Writing '1' disables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **DDRPERFMLPEN**: DDRPERFM APB clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep

1: Writing '1' disables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **LTDCLPEN**: LTDC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.180 RCC APB5 sleep clock enable for MCU set register (RCC_MC_APB5LPENSETR)

Address offset: 0x388

Reset value: 0x0011 391D

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN
										rs	rs				rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
		rs	rs	rs			rs				rs	rs	rs		rs

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENSTPEN**: STGEN Controller part, peripheral clocks enable during CStop mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop
1: Writing '1' enables the peripheral clocks in CStop, reading '1' means that the peripheral clocks are enabled in CStop

Bit 20 **STGENLPEN**: STGEN Controller part, peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECLPEN**: BSEC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **TZPCLPEN**: TZPC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 **TZC2LPEN**: TZC AXI port 2 peripheral clocks enable during CSleep mode
Set by software. See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2 in CSleep
1: Writing '1' enables the pclk5 and aclk_tzc2 clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 2 in CSleep

Bit 11 **TZC1LPEN**: TZC AXI port 1 peripheral clocks enable during CSleep mode
Set by software. See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1 in CSleep
1: Writing '1' enables the pclk5 and aclk_tzc1 clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 1 in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBLPEN**: RTC APB clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1LPEN**: USART1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **I2C6LPEN**: I2C6 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **I2C4LPEN**: I2C4 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6LPEN**: SPI6 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.181 RCC APB5 sleep clock enable for MPU set register (RCC_MP_APB5LPENSETR)

Address offset: 0x308

Reset value: 0x0011 B91D

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN
										rs	rs				rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG1APBLPEN	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
rs		rs	rs	rs			rs				rs	rs	rs		rs

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENSTPEN**: STGEN Controller part, peripheral clocks enable during CStop mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop

1: Writing '1' enables the peripheral clocks in CStop, reading '1' means that the peripheral clocks are enabled in CStop

Bit 20 **STGENLPEN**: STGEN Controller part, peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECLPEN**: BSEC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **IWDG1APBLPEN**: IWDG1 APB clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep

1: Writing '1' enables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bit 14 Reserved, must be kept at reset value.

Bit 13 **TZPCLPEN**: TZPC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 **TZC2LPEN**: TZC AXI port 2 peripheral clocks enable during CSleep mode

Set by software. See *Section : Clock distribution for TZC* for details.

0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2 in CSleep

1: Writing '1' enables the pclk5 and aclk_tzc2 clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 2 in CSleep

Bit 11 **TZC1LPEN**: TZC AXI port 1 peripheral clocks enable during CSleep mode

Set by software. See *Section : Clock distribution for TZC* for details.

0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1 in CSleep

1: Writing '1' enables the pclk5 and aclk_tzc1 clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 1 in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBLPEN**: RTC APB clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1LPEN**: USART1 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **I2C6LPEN**: I2C6 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **I2C4LPEN**: I2C4 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6LPEN**: SPI6 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.182 RCC APB5 sleep clock enable for MCU clear register (RCC_MC_APB5LPENCLRR)

Address offset: 0x38C

Reset value: 0x0011 391D

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN
										rc_w1	rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
		rc_w1	rc_w1	rc_w1			rc_w1				rc_w1	rc_w1	rc_w1		rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENSTPEN**: STGEN peripheral clocks enable during CStop mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop

1: Writing '1' disables the peripheral clocks in CStop, reading '1' means that the peripheral clocks are enabled in CStop

Bit 20 **STGENLPEN**: STGEN peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECLPEN**: BSEC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **TZPCLPEN**: TZPC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 **TZC2LPEN**: TZC AXI port 2 peripheral clocks disable during CSleep mode
Cleared by software. See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2 in CSleep
1: Writing '1' disables the AXI port 2 peripheral clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 2 in CSleep

Bit 11 **TZC1LPEN**: TZC AXI port 1 peripheral clocks disable during CSleep mode
Cleared by software. See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1 in CSleep
1: Writing '1' disables the AXI port 1 peripheral clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 1 in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBLPEN**: RTC APB clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1LPEN**: USART1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **I2C6LPEN**: I2C6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **I2C4LPEN**: I2C4 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6LPEN**: SPI6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.183 RCC APB5 sleep clock enable for MPU clear register (RCC_MP_APB5LPENCLRR)

Address offset: 0x30C

Reset value: 0x0011 B91D

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN
										rc_w1	rc_w1				rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG1APBLPEN	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
rc_w1		rc_w1	rc_w1	rc_w1			rc_w1				rc_w1	rc_w1	rc_w1		rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **STGENSTPEN**: STGEN peripheral clocks enable during CStop mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CStop

1: Writing '1' disables the peripheral clocks in CStop, reading '1' means that the peripheral clocks are enabled in CStop

Bit 20 **STGENLPEN**: STGEN peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BSECLPEN**: BSEC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 **IWDG1APBLPEN**: IWDG1 APB clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the APB clock is disabled in CSleep

1: Writing '1' disables the APB clock in CSleep, reading '1' means that the APB clock is enabled in CSleep

Bit 14 Reserved, must be kept at reset value.

Bit 13 **TZPCLPEN**: TZPC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 12 **TZC2LPEN**: TZC AXI port 2 peripheral clocks disable during CSleep mode
Cleared by software. See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 2 in CSleep
1: Writing '1' disables the AXI port 2 peripheral clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 2 in CSleep

Bit 11 **TZC1LPEN**: TZC AXI port 1 peripheral clocks disable during CSleep mode
Cleared by software. See *Section : Clock distribution for TZC* for details.
0: Writing '0' has no effect, reading '0' means that the clocks are disabled for AXI port 1 in CSleep
1: Writing '1' disables the AXI port 1 peripheral clocks in CSleep, reading '1' means that the clocks are enabled for AXI port 1 in CSleep

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **RTCAPBLPEN**: RTC APB clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **USART1LPEN**: USART1 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **I2C6LPEN**: I2C6 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **I2C4LPEN**: I2C4 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SPI6LPEN**: SPI6 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.184 RCC AHB5 peripheral enable for MPU set register (RCC_MP_AHB5LPENSETR)

Address offset: 0x310

Reset value: 0x0000 0171

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
							rs		rs	rs	rs				rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMLPEN**: BKPSRAM clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the clock is disabled in CSleep

1: Writing '1' enables the peripheral clock in CSleep, reading '1' means that the clock is enabled in CSleep

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1LPEN**: RNG1 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH1LPEN**: HASH1 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP1LPEN**: CRYP1 (3DES/AES1) peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZLPEN**: GPIOZ Secure peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.185 RCC AHB5 peripheral enable for MCU set register (RCC_MC_AHB5LPENSETR)

Address offset: 0x390

Reset value: 0x0000 0171

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
							rs		rs	rs	rs				rs

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMLPEN**: BKPSRAM clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the clock is disabled in CSleep
1: Writing '1' enables the peripheral clock in CSleep, reading '1' means that the clock is enabled in CSleep

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1LPEN**: RNG1 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH1LPEN**: HASH1 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP1LPEN**: CRYP1 (3DES/AES1) peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZLPEN**: GPIOZ Secure peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.186 RCC AHB5 sleep clock enable for MPU clear register (RCC_MP_AHB5LPENCLRR)

Address offset: 0x314

Reset value: 0x0000 0171

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
							rc_w1		rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMLPEN**: BKPSRAM clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the clock is disabled in CSleep

1: Writing '1' disables the clock in CSleep, reading '1' means that the clock is enabled in CSleep

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1LPEN**: RNG1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH1LPEN**: HASH1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP1LPEN**: CRYP1 (3DES/AES1) peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZLPEN**: GPIOZ Secure peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.187 RCC AHB5 sleep clock enable for MCU clear register (RCC_MC_AHB5LPENCLRR)

Address offset: 0x394

Reset value: 0x0000 0171

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
							rc_w1		rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **BKPSRAMLPEN**: BKPSRAM clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the clock is disabled in CSleep

1: Writing '1' disables the clock in CSleep, reading '1' means that the clock is enabled in CSleep

Bit 7 Reserved, must be kept at reset value.

Bit 6 **RNG1LPEN**: RNG1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH1LPEN**: HASH1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP1LPEN**: CRYP1 (3DES/AES1) peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **GPIOZLPEN**: GPIOZ Secure peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.188 RCC AHB6 sleep clock enable for MPU set register (RCC_MP_AHB6LPENSETR)

Address offset: 0x318

Reset value: 0x0113 57A1

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN
							rs				rs			rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	rs		rs	rs	rs	rs	rs	rs							rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHLPEN**: USBH peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1LPEN**: CRC1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2LPEN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPILPEN**: QUADSPI peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCLPEN**: FMC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **ETHSTPEN**: ETH peripheral clock enable during CStop mode

Allows the ETH block to receive the clock coming from pads ETH_TX_CLK and ETH_RX_CLK during CStop.

Set by software.

0: Writing '0' has no effect, reading '0' means that the ETH RX and TX kernel clocks are gated in CStop

1: Writing '1' enables the ETH RX and TX kernel clocks in CStop, reading '1' means that the ETH RX and TX kernel clocks are enabled in CStop

Bit 10 **ETHMACLPEN**: Enable of the bus interface clocks for ETH block during CSleep mode

The bus interface clocks for ETH are hclk6 and aclk.

Set by software.

0: Writing '0' has no effect, reading '0' means that the bus interface clocks are disabled in CSleep

1: Writing '1' enables the bus interface clocks in CSleep, reading '1' means that the bus interface clocks are enabled in CSleep

Bit 9 **ETHRXLPEN**: Enable of the Ethernet Reception Clock during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the reception clock is disabled in CSleep

1: Writing '1' enables the reception clock in CSleep, reading '1' means that the reception clock is enabled in CSleep

Bit 8 **ETHTXLPEN**: Enable of the Ethernet Transmission Clock during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled in CSleep

1: Writing '1' enables the transmission clock in CSleep, reading '1' means that the transmission clock is enabled in CSleep

Bit 7 **ETHCKLPEN**: Enable of the Ethernet clock generated by the RCC (eth_ker_ck) during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled in CSleep

1: Writing '1' enables the eth_ker_ck clock in CSleep, reading '1' means that the eth_ker_ck clock is enabled in CSleep

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.189 RCC AHB6 sleep clock enable for MCU set register (RCC_MC_AHB6LPENSETR)

Address offset: 0x398

Reset value: 0x0113 57A1

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN
							rs				rs			rs	rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	rs		rs	rs	rs	rs	rs	rs							rs

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHLPEN**: USBH peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1LPEN**: CRC1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2LPEN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPILPEN**: QUADSPI peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCLPEN**: FMC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **ETHSTPEN**: ETH peripheral clock enable during CStop mode
Allows the ETH block to receive the clock coming from pads ETH_TX_CLK and ETH_RX_CLK during CStop.
Set by software.
0: Writing '0' has no effect, reading '0' means that the ETH RX and TX kernel clocks are gated in CStop
1: Writing '1' enables the ETH RX and TX kernel clocks in CStop, reading '1' means that the ETH RX and TX kernel clocks are enabled in CStop

Bit 10 **ETHMACLPEN**: Enable of the bus interface clocks for ETH block during CSleep mode
The bus interface clocks for ETH are hclk6 and aclk.
Set by software.
0: Writing '0' has no effect, reading '0' means that the bus interface clocks are disabled in CSleep
1: Writing '1' enables the bus interface clocks in CSleep, reading '1' means that the bus interface clocks are enabled in CSleep

Bit 9 **ETHRXLPEN**: Enable of the Ethernet Reception Clock during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the reception clock is disabled in CSleep
1: Writing '1' enables the reception clock in CSleep, reading '1' means that the reception clock is enabled in CSleep

Bit 8 **ETHTXLPEN**: Enable of the Ethernet Transmission Clock during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled in CSleep
1: Writing '1' enables the transmission clock in CSleep, reading '1' means that the transmission clock is enabled in CSleep

Bit 7 **ETHCKLPEN**: Enable of the Ethernet clock generated by the RCC (eth_ker_ck) during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled in CSleep
1: Writing '1' enables the eth_ker_ck clock in CSleep, reading '1' means that the eth_ker_ck clock is enabled in CSleep

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.190 RCC AHB6 sleep clock enable for MPU clear register (RCC_MP_AHB6LPENCLRR)

Address offset: 0x31C

Reset value: 0x0113 57A1

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN
							rc_w1				rc_w1			rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1							rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHLPEN**: USBH peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1LPEN**: CRC1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2LPEN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPILPEN**: QUADSPI peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCLPEN**: FMC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **ETHSTPEN**: ETH peripheral clock enable during CStop mode

Allows the ETH block to receive the clock coming from pads ETH_TX_CLK and ETH_RX_CLK during CStop.

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the ETH RX and TX kernel clocks are gated in CStop

1: Writing '1' disabled the ETH RX and TX kernel clocks in CStop, reading '1' means that the ETH RX and TX kernel clocks are enabled in CStop

Bit 10 **ETHMACLPEN**: Disable of the bus interface clocks for ETH block during CSleep mode

The bus interface clocks for ETH are hclk6 and aclk.

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the bus interface clocks are disabled in CSleep

1: Writing '1' disables the bus interface clocks in CSleep, reading '1' means that the bus interface clocks are enabled in CSleep

Bit 9 **ETHRXLPEN**: Disable of the Ethernet Reception Clock during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the reception clock is disabled in CSleep

1: Writing '1' disables the reception clock in CSleep, reading '1' means that the reception clock is enabled in CSleep

Bit 8 **ETHTXLPEN**: Disable of the Ethernet Transmission Clock during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled in CSleep

1: Writing '1' disables the transmission clock in CSleep, reading '1' means that the transmission clock is enabled in CSleep

Bit 7 **ETHCKLPEN**: Disable of the Ethernet clock generated by the RCC (eth_ker_ck) during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled in CSleep

1: Writing '1' disables the eth_ker_ck clock in CSleep, reading '1' means that the eth_ker_ck clock is enabled in CSleep

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.191 RCC AHB6 sleep clock enable for MCU clear register (RCC_MC_AHB6LPENCLRR)

Address offset: 0x39C

Reset value: 0x0113 57A1

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN
							rc_w1				rc_w1			rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1							rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBHLPEN**: USBH peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CRC1LPEN**: CRC1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **SDMMC2LPEN**: SDMMC2 and SDMMC2 delay (DLYBSD2) block peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 16 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay (DLYBSD1) block peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 15 Reserved, must be kept at reset value.

Bit 14 **QSPILPEN**: QUADSPI peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 13 Reserved, must be kept at reset value.

Bit 12 **FMCLPEN**: FMC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **ETHSTPEN**: ETH peripheral clock enable during CStop mode
Allows the ETH block to receive the clock coming from pads ETH_TX_CLK and ETH_RX_CLK during CStop.
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the ETH RX and TX kernel clocks are gated in CStop
1: Writing '1' disabled the ETH RX and TX kernel clocks in CStop, reading '1' means that the ETH RX and TX kernel clocks are enabled in CStop

Bit 10 **ETHMACLPEN**: Disable of the bus interface clocks for ETH block during CSleep mode
The bus interface clocks for ETH are hclk6 and aclk.
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the bus interface clocks are disabled in CSleep
1: Writing '1' disables the bus interface clocks in CSleep, reading '1' means that the bus interface clocks are enabled in CSleep

Bit 9 **ETHRXLPEN**: Disable of the Ethernet Reception Clock during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the reception clock is disabled in CSleep
1: Writing '1' disables the reception clock in CSleep, reading '1' means that the reception clock is enabled in CSleep

Bit 8 **ETHTXLPEN**: Disable of the Ethernet Transmission Clock during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the transmission clock is disabled in CSleep
1: Writing '1' disables the transmission clock in CSleep, reading '1' means that the transmission clock is enabled in CSleep

Bit 7 **ETHCKLPEN**: Disable of the Ethernet clock generated by the RCC (eth_ker_ck) during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that eth_ker_ck clock is disabled in CSleep
1: Writing '1' disables the eth_ker_ck clock in CSleep, reading '1' means that the eth_ker_ck clock is enabled in CSleep

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.192 RCC AHB6 secure sleep clock enable for MPU set register (RCC_MP_TZAHB6LPENSETR)

Address offset: 0x320

Reset value: 0x0000 0001

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA peripheral clocks enable during CSleep mode

Set by software.

Note that this bit is “ORed” with the MDMALPEN bits of RCC_MP_AHB6LPENSETR/CLRR and RCC_MC_AHB6LPENSETR/CLRR registers.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.193 RCC AHB6 secure sleep clock enable for MPU clear register (RCC_MP_TZAHB6LPENCLRR)

Address offset: 0x324

Reset value: 0x0000 0001

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **MDMALPEN**: MDMA peripheral clocks enable during CSleep mode

Cleared by software.

Note that this bit is “ORed” with the MDMALPEN bits of RCC_MP_AHB6LPENSETR/CLRR and RCC_MC_AHB6LPENSETR/CLRR registers.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.194 RCC AHB2 sleep clock enable for MPU set register (RCC_MP_AHB2LPENSETR)

Address offset: 0xB18

Reset value: 0x0001 0127

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN
															rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
							rs			rs			rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3LPEN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOLPEN**: USBO peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12LPEN**: ADC1 and 2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXLPEN**: DMAMUX peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **DMA2LPEN**: DMA2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **DMA1LPEN**: DMA1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.195 RCC AHB2 sleep clock enable for MCU set register (RCC_MC_AHB2LPENSETR)

Address offset: 0xB98

Reset value: 0x0001 0127

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN
															rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
							rs			rs			rs	rs	rs

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3LPEN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOLPEN**: USBO peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12LPEN**: ADC1 and 2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXLPEN**: DMAMUX peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **DMA2LPEN**: DMA2 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **DMA1LPEN**: DMA1 peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.196 RCC AHB2 sleep clock enable for MPU clear register (RCC_MP_AHB2LPENCLRR)

Address offset: 0xB1C

Reset value: 0x0001 0127

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN
															rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
							rc_w1			rc_w1			rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3LPEN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOLPEN**: USBO peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12LPEN**: ADC1 and 2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXLPEN**: DMAMUX peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **DMA2LPEN**: DMA2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **DMA1LPEN**: DMA1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.197 RCC AHB2 sleep clock enable for MCU clear register (RCC_MC_AHB2LPENCLRR)

Address offset: 0xB9C

Reset value: 0x0001 0127

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN
															rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
							rc_w1			rc_w1			rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SDMMC3LPEN**: SDMMC3 and SDMMC3 delay (DLYBSD3) block peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **USBOLPEN**: USBO peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **ADC12LPEN**: ADC1 and 2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **DMAMUXLPEN**: DMAMUX peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **DMA2LPEN**: DMA2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **DMA1LPEN**: DMA1 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.198 RCC AHB3 sleep clock enable for MPU set register (RCC_MP_AHB3LPENSETR)

Address offset: 0xB20

Reset value: 0x0000 18F1

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
			rs	rs				rs	rs	rs	rs				rs

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCLPEN**: IPCC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **HSEMLPEN**: HSEM peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2LPEN**: CRC2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **RNG2LPEN**: RNG2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH2LPEN**: HASH2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP2LPEN**: CRYP2 (3DES/AES2) peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMILPEN**: DCMI peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.199 RCC AHB3 sleep clock enable for MCU set register (RCC_MC_AHB3LPENSETR)

Address offset: 0xBA0

Reset value: 0x0000 18F1

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
			rs	rs				rs	rs	rs	rs				rs

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCLPEN**: IPCC peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **HSEMLPEN**: HSEM peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2LPEN**: CRC2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **RNG2LPEN**: RNG2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH2LPEN**: HASH2 peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP2LPEN**: CRYP2 (3DES/AES2) peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMILPEN**: DCMI peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.200 RCC AHB3 sleep clock enable for MPU clear register (RCC_MP_AHB3LPENCLRR)

Address offset: 0xB24

Reset value: 0x0000 18F1

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
			rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCLPEN**: IPCC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **HSEMLPEN**: HSEM peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2LPEN**: CRC2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **RNG2LPEN**: RNG2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH2LPEN**: HASH2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP2LPEN**: CRYP2 (3DES/AES2) peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMILPEN**: DCMI peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.201 RCC AHB3 sleep clock enable for MCU clear register (RCC_MC_AHB3LPENCLRR)

Address offset: 0xBA4

Reset value: 0x0000 18F1

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
			rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1				rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **IPCCLPEN**: IPCC peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 11 **HSEMLPEN**: HSEM peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **CRC2LPEN**: CRC2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **RNG2LPEN**: RNG2 peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **HASH2LPEN**: HASH2 peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **CRYP2LPEN**: CRYP2 (3DES/AES2) peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DCMILPEN**: DCMI peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.202 RCC AHB4 sleep clock enable for MPU set register (RCC_MP_AHB4LPENSETR)

Address offset: 0xB28

Reset value: 0x0000 07FF

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
					rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKLPEN**: GPIOK peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **GPIOJLPEN**: GPIOJ peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **GPIOILPEN**: GPIOI peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **GPIOHLPEN**: GPIOH peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **GPIOGLPEN**: GPIOG peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **GPIOFLPEN**: GPIOF peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **GPIOELPEN**: GPIOE peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **GPIODLPEN**: GPIOD peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **GPIOCLPEN**: GPIOC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **GPIOBLPEN**: GPIOB peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **GPIOALPEN**: GPIOA peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.203 RCC AHB4 sleep clock enable for MCU set register (RCC_MC_AHB4LPENSETR)

Address offset: 0xBA8

Reset value: 0x0000 07FF

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
					rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKLPEN**: GPIOK peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **GPIOJLPEN**: GPIOJ peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **GPIOILPEN**: GPIOI peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **GPIOHLPEN**: GPIOH peripheral clocks enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **GPIOGLPEN**: GPIOG peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **GPIOFLPEN**: GPIOF peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **GPIOELPEN**: GPIOE peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **GPIODLPEN**: GPIOD peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **GPIOCLPEN**: GPIOC peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **GPIOBLPEN**: GPIOB peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **GPIOALPEN**: GPIOA peripheral clocks enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' enables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.204 RCC AHB4 sleep clock enable for MPU clear register (RCC_MP_AHB4LPENCLRR)

Address offset: 0xB2C

Reset value: 0x0000 07FF

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKLPEN**: GPIOK peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **GPIOJLPEN**: GPIOJ peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **GPIOILPEN**: GPIOI peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **GPIOHLPEN**: GPIOH peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **GPIOGLPEN**: GPIOG peripheral clocks enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep

1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **GPIOFLPEN**: GPIOF peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **GPIOELPEN**: GPIOE peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **GPIODLPEN**: GPIOD peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **GPIOCLPEN**: GPIOC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **GPIOBLPEN**: GPIOB peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **GPIOALPEN**: GPIOA peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.205 RCC AHB4 sleep clock enable for MCU clear register (RCC_MC_AHB4LPENCLRR)

Address offset: 0xBAC

Reset value: 0x0000 07FF

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **GPIOKLPEN**: GPIOK peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 9 **GPIOJLPEN**: GPIOJ peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 8 **GPIOILPEN**: GPIOI peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 7 **GPIOHLPEN**: GPIOH peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 6 **GPIOGLPEN**: GPIOG peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 5 **GPIOFLPEN**: GPIOF peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 4 **GPIOELPEN**: GPIOE peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 3 **GPIODLPEN**: GPIOD peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 2 **GPIOCLPEN**: GPIOC peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 1 **GPIOBLPEN**: GPIOB peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

Bit 0 **GPIOALPEN**: GPIOA peripheral clocks enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the peripheral clocks are disabled in CSleep
1: Writing '1' disables the peripheral clocks in CSleep, reading '1' means that the peripheral clocks are enabled in CSleep

10.7.206 RCC AXI sleep clock enable for MPU set register (RCC_MP_AXIMLPENSETR)

Address offset: 0xB30

Reset value: 0x0000 0001

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRAMLPEN**: SYSRAM interface clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface clock is enabled in CSleep

10.7.207 RCC AXI sleep clock enable for MCU set register (RCC_MC_AXIMLPENSETR)

Address offset: 0xBB0

Reset value: 0x0000 0001

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
															rs

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRAMLPEN**: SYSRAM interface clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface clock is enabled in CSleep

10.7.208 RCC AXI sleep clock enable for MPU clear register (RCC_MP_AXIMLPENCLRR)

Address offset: 0xB34

Reset value: 0x0000 0001

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRAMLPEN**: SYSRAM interface clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the memory interface is disabled in CSleep

1: Writing '1' disables the memory interface in CSleep, reading '1' means that the memory interface is enabled in CSleep

10.7.209 RCC AXI sleep clock enable for MCU clear register (RCC_MC_AXIMLPENCLRR)

Address offset: 0xBB4

Reset value: 0x0000 0001

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
															rc_w1

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSRAMLPEN**: SYSRAM interface clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the memory interface is disabled in CSleep

1: Writing '1' disables the memory interface in CSleep, reading '1' means that the memory interface is enabled in CSleep

10.7.210 RCC MLAHB sleep clock enable for MPU set register (RCC_MP_MLAHBLPENSETR)

Address offset: 0xB38

Reset value: 0x0000 0017

This register is used by the MPU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
											rs		rs	rs	rs

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMLPEN**: RETRAM interface clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAM34LPEN**: SRAM3 and SRAM4 interface clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 1 **SRAM2LPEN**: SRAM2 interface clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep.

Bit 0 **SRAM1LPEN**: SRAM1 interface clock enable during CSleep mode

Set by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

10.7.211 RCC MLAHB sleep clock enable for MCU set register (RCC_MC_MLAHBLPENSETR)

Address offset: 0xBB8

Reset value: 0x0000 0017

This register is used by the MCU in order to set the PERxLPEN bit of the corresponding peripheral to '1'. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
											rs		rs	rs	rs

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMLPEN**: RETRAM interface clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAM34LPEN**: SRAM3 and SRAM4 interface clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 1 **SRAM2LPEN**: SRAM2 interface clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep.

Bit 0 **SRAM1LPEN**: SRAM1 interface clock enable during CSleep mode
Set by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' enables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

10.7.212 RCC MLAHB sleep clock enable for MPU clear register (RCC_MP_MLAHBLPENCLRR)

Address offset: 0xB3C

Reset value: 0x0000 0017

This register is used by the MPU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
											rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMLPEN**: RETRAM interface clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAM34LPEN**: SRAM3 and SRAM4 interface clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 1 **SRAM2LPEN**: SRAM2 interface clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 0 **SRAM1LPEN**: SRAM1 interface clock enable during CSleep mode

Cleared by software.

0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep

1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

10.7.213 RCC MLAHB sleep clock enable for MCU clear register (RCC_MC_MLAHBLPENCLRR)

Address offset: 0xBBC

Reset value: 0x0000 0017

This register is used by the MCU in order to clear the PERxLPEN bit of the corresponding peripheral. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
											rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RETRAMLPEN**: RETRAM interface clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SRAM34LPEN**: SRAM3 and SRAM4 interface clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 1 **SRAM2LPEN**: SRAM2 interface clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

Bit 0 **SRAM1LPEN**: SRAM1 interface clock enable during CSleep mode
Cleared by software.
0: Writing '0' has no effect, reading '0' means that the memory interface clock is disabled in CSleep
1: Writing '1' disables the memory interface clock in CSleep, reading '1' means that the memory interface is enabled in CSleep

10.7.214 RCC BOOTROM reset status clear register (RCC_BR_RSTSCLRR)

Address offset: 0x400

Reset value: 0x0000 0015

This register is used by the BOOTROM to check the reset source. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' clears the corresponding bit to '0'. In order to identify the reset source, the MPU application must use *RCC MPU reset status clear register (RCC_MP_RSTSCLRR)*, and the MCU application must use the *RCC MCU reset status clear register (RCC_MC_RSTSCLRR)*. Refer to *Section 10.3.13: Reset source identification* for details.

This register except MPUP[1:0]RSTF flags is located into V_{DD} domain, and is reset by por_rst reset. The MPUP[1:0]RSTF flags are located into V_{DDCORE} and are reset by nreset. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	MPUP1RSTF	MPUP0RSTF	Res.	Res.	Res.	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **MPUP1RSTF**: MPU processor 1 reset flag

Cleared by software, set by hardware when a MPU processor 1 reset occurred.

0: Writing '0' has no effect, reading '0' means that no MPU processor 1 reset occurred (default after nreset reset)

1: Writing '1' clears the MPUP1RSTF flag, reading '1' means that MPU processor 1 reset occurred

Bit 13 **MPUP0RSTF**: MPU processor 0 reset flag

Cleared by software, set by hardware when a MPU processor 0 reset occurred.

0: Writing '0' has no effect, reading '0' means that no MPU processor 0 reset occurred (default after nreset reset)

1: Writing '1' clears the MPUP0RSTF flag, reading '1' means that MPU processor 0 reset occurred

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **IWDG2RSTF**: IWDG2 reset flag

Cleared by software, set by hardware when a IWDG2 reset occurred.

0: Writing '0' has no effect, reading '0' means that no IWDG2 reset occurred (default after por_rst reset)

1: Writing '1' clears the IWDG2RSTF flag, reading '1' means that a IWDG2 reset occurred

Bit 8 **IWDG1RSTF**: IWDG1 reset flag

Cleared by software, set by hardware when a IWDG1 reset occurred.

0: Writing '0' has no effect, reading '0' means that no IWDG1 reset occurred (default after por_rst reset)

1: Writing '1' clears the IWDG1RSTF flag, reading '1' means that a IWDG1 reset occurred

Bit 7 **MCSYSRSTF**: MCU System reset flag

Cleared by software, set by hardware when a MCU system reset occurred.

Note that for the RCC_BR_RSTSCLRR register, this bit is forced to '0' when the option byte OTP_MCU_SYSRST_EN does not allow the MCU to perform a system reset.

0: Writing '0' has no effect, reading '0' means that no system reset generated by the MCU occurred (default after por_rst reset)

1: Writing '1' clears the MCURSTF flag, reading '1' means that a system reset generated by the MCU occurred

Bit 6 **MPSYSRSTF**: MPU System reset flag

Cleared by software, set by hardware when a MPU system reset occurred.

0: Writing '0' has no effect, reading '0' means that no system reset generated by the MPU occurred (default after por_rst reset)

1: Writing '1' clears the MCURSTF flag, reading '1' means that a system reset generated by the MPU occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag

Cleared by software, set by hardware when a reset occurred because V_{DDCORE} is lower than the expected value.

0: Writing '0' has no effect, reading '0' means that V_{DDCORE} is not the origin of the reset

1: Writing '1' clears the VCORERSTF flag, reading '1' means that V_{DDCORE} is the origin of the reset (default after por_rst reset)

Bit 3 **HCSSRSTF**: HSE CSS reset flag

Cleared by software, set by hardware when a failure is detected on HSE.

0: Writing '0' has no effect, reading '0' means that no HSE CSS reset occurred (default after por_rst reset)

1: Writing '1' clears the HCSSRSTF flag, reading '1' means that a HSE CSS reset occurred

Bit 2 **PADRSTF**: NRST reset flag

Cleared by software, set by hardware when a PAD reset occurred.

0: Writing '0' has no effect, reading '0' means that no PAD reset occurred (default after por_rst reset)

1: Writing '1' clears the PADRSTF flag, reading '1' means that a PAD reset occurred

Bit 1 **BORRSTF**: BOR reset flag

Cleared by software, set by hardware when a BOR reset occurred.

0: Writing '0' has no effect, reading '0' means that no BOR reset occurred (default after por_rst reset)

1: Writing '1' clears the BORRSTF flag, reading '1' means that a BOR reset occurred

Bit 0 **PORRSTF**: POR/PDR reset flag

Cleared by software, set by hardware when a POR/PDR reset occurred.

0: Writing '0' has no effect, reading '0' means that no POR/PDR reset occurred

1: Writing '1' clears the PORRSTF flag, reading '1' means that a POR/PDR reset occurred (default after por_rst reset)

10.7.215 RCC MCU reset status clear register (RCC_MC_RSTSCLRR)

Address offset: 0xC00

Reset value: 0x0000 0015

This register is used by the MCU to check the reset source. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' clears the corresponding bit to '0' .Refer to *Section 1.3.12: Reset Source Identification* for details.This register is located into V_{DD} domain, and is reset by por_rst reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	WWDG1RSTF	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	MCURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **WWDG1RSTF**: WWDG1 reset flag

Cleared by software, set by hardware when a WWDG1 reset occurred.

0: Writing '0' has no effect, reading '0' means that no WWDG1 reset occurred (default after por_rst reset)

1: Writing '1' clears the WWDG1RSTF flag, reading '1' means that a WWDG1 reset occurred

Bit 9 **IWDG2RSTF**: IWDG2 reset flag

Cleared by software, set by hardware when a IWDG2 reset occurred.

0: Writing '0' has no effect, reading '0' means that no IWDG2 reset occurred (default after por_rst reset)

1: Writing '1' clears the IWDG2RSTF flag, reading '1' means that a IWDG2 reset occurred

Bit 8 **IWDG1RSTF**: IWDG1 reset flag

Cleared by software, set by hardware when a IWDG1 reset occurred.

0: Writing '0' has no effect, reading '0' means that no IWDG1 reset occurred (default after por_rst reset)

1: Writing '1' clears the IWDG1RSTF flag, reading '1' means that a IWDG1 reset occurred

Bit 7 **MCSYSRSTF**: MCU System reset flag

Cleared by software, set by hardware when a MCU system reset occurred.

Note that for the RCC_BR_RSTSCLRR register, this bit is forced to '0' when the option byte OTP_MCU_SYSRST_EN does not allow the MCU to perform a system reset.

0: Writing '0' has no effect, reading '0' means that no system reset generated by the MCU occurred (default after por_rst reset)

1: Writing '1' clears the MCURSTF flag, reading '1' means that a system reset generated by the MCU occurred

Bit 6 **MPSYSRSTF**: MPU System reset flag

Cleared by software, set by hardware when a MPU system reset occurred.

0: Writing '0' has no effect, reading '0' means that no system reset generated by the MPU occurred (default after por_rst reset)

1: Writing '1' clears the MCURSTF flag, reading '1' means that a system reset generated by the MPU occurred

Bit 5 **MCURSTF**: MCU reset flag

Cleared by software, set by hardware when a MCU reset occurred.

0: Writing '0' has no effect, reading '0' means that no MCU reset occurred (default after por_rst reset)

1: Writing '1' clears the MCURSTF flag, reading '1' means that a MCU reset occurred

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag

Cleared by software, set by hardware when a reset occurred because V_{DDCORE} is lower than the expected value.

0: Writing '0' has no effect, reading '0' means that V_{DDCORE} is not the origin of the reset

1: Writing '1' clears the VCORERSTF flag, reading '1' means that V_{DDCORE} is the origin of the reset (default after por_rst reset)

Bit 3 **HCSSRSTF**: HSE CSS reset flag

Cleared by software, set by hardware when a failure is detected on HSE.

0: Writing '0' has no effect, reading '0' means that no HSE CSS reset occurred (default after por_rst reset)

1: Writing '1' clears the HCSSRSTF flag, reading '1' means that a HSE CSS reset occurred

Bit 2 **PADRSTF**: NRST reset flag

Cleared by software, set by hardware when a PAD reset occurred.

0: Writing '0' has no effect, reading '0' means that no PAD reset occurred (default after por_rst reset)

1: Writing '1' clears the PADRSTF flag, reading '1' means that a PAD reset occurred

Bit 1 **BORRSTF**: BOR reset flag

Cleared by software, set by hardware when a BOR reset occurred.

0: Writing '0' has no effect, reading '0' means that no BOR reset occurred (default after por_rst reset)

1: Writing '1' clears the BORRSTF flag, reading '1' means that a BOR reset occurred

Bit 0 **PORRSTF**: POR/PDR reset flag

Cleared by software, set by hardware when a POR/PDR reset occurred.

0: Writing '0' has no effect, reading '0' means that no POR/PDR reset occurred

1: Writing '1' clears the PORRSTF flag, reading '1' means that a POR/PDR reset occurred (default after por_rst reset)

10.7.216 RCC MPU reset status clear register (RCC_MP_RSTSCLRR)

Address offset: 0x408

Reset value: 0x0000 0000

This register is used by the MPU to check the reset source. This register is updated by the BOOTROM code, after a power-on reset (por_rst), a system reset (nreset), or an exit from Standby or CStandby.

Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' clears the corresponding bit to '0'.

Refer to *Section 10.3.13: Reset source identification* for details.

The register is located in V_{DDCORE}.

If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPARE	MPUP1RSTF	MPUP0RSTF	CSTDBYRSTF	STDBYRSTF	Res.	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SPARE**: Spare bits

Set and reset by software. Reserved for future use.

Bit 14 **MPUP1RSTF**: MPU processor 1 reset flag

Cleared by software, set by the BOOTROM code when a MPU processor 1 reset occurred.

0: No MPU processor 1 reset occurred (default after nreset reset)
1: MPU processor 1 reset occurred

Bit 13 **MPUP0RSTF**: MPU processor 0 reset flag

Cleared by software, set by the BOOTROM code when a MPU processor 0 reset occurred.

0: No MPU processor 0 reset occurred (default after nreset reset)
1: MPU processor 0 reset occurred

Bit 12 **CSTDBYRSTF**: MPU CStandby reset flag

Cleared by software, set by the BOOTROM code when the MPU exits from CStandby.

0: MPU has not been in CStandby mode
1: MPU has been in CStandby mode

Bit 11 **STDBYRSTF**: System Standby reset flag

Cleared by software, set by the BOOTROM code when exiting from system Standby.

0: System has not been in Standby mode
1: System has been in Standby mode

Bit 10 Reserved, must be kept at reset value.

Bit 9 **IWDG2RSTF**: IWDG2 reset flag

Cleared by software, set by the BOOTROM code when a IWDG2 reset occurred.

0: No IWDG2 reset occurred
1: An IWDG2 reset occurred

Bit 8 **IWDG1RSTF**: IWDG1 reset flag

Cleared by software, set by the BOOTROM code when a IWDG1 reset occurred.

0: No IWDG1 reset occurred
1: An IWDG1 reset occurred

Bit 7 **MCSYSRSTF**: MCU System reset flag

Cleared by software, set by the BOOTROM code when a MCU system reset occurred.

0: No system reset generated by the MCU occurred
1: A system reset generated by the MCU occurred

Bit 6 **MPSYSRSTF**: MPU System reset flag
Cleared by software, set by the BOOTROM code when a MPU system reset occurred.
0: No system reset generated by the MPU occurred
1: A system reset generated by the MPU occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag
Cleared by software, set by the BOOTROM code when a reset occurred because V_{DDCORE} is lower than the expected value.
0: V_{DDCORE} is not the origin of the reset
1: V_{DDCORE} is the origin of the reset

Bit 3 **HCSSRSTF**: HSE CSS reset flag
Cleared by software, set by the BOOTROM code when a failure is detected on HSE.
0: No HSE CSS reset occurred
1: A HSE CSS reset occurred

Bit 2 **PADRSTF**: NRST reset flag
Cleared by software, set by the BOOTROM code when a PAD reset occurred.
0: No PAD reset occurred
1: A PAD reset occurred

Bit 1 **BORRSTF**: BOR reset flag
Cleared by software, set by the BOOTROM code when a BOR reset occurred.
0: No BOR reset occurred
1: A BOR reset occurred

Bit 0 **PORRSTF**: POR/PDR reset flag
Cleared by software, set by the BOOTROM code when a POR/PDR reset occurred.
0: No POR/PDR reset occurred
1: A POR/PDR reset occurred

10.7.217 RCC MPU reset status set register (RCC_MP_RSTSSETR)

Address offset: 0x420

Reset value: 0x0000 0000

This register is dedicated to the BOOTROM code in order to update the reset source. This register is updated by the BOOTROM code, after a power-on reset (por_rst), a system reset (nreset), or an exit from Standby or CStandby. The application software shall not use this register. In order to identify the reset source, the MPU application must use *RCC MPU reset status clear register (RCC_MP_RSTSCLRR)*, and the MCU application must use the *RCC MCU reset status clear register (RCC_MC_RSTSCLRR)*.

Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'.

Refer to *Section 10.3.13: Reset source identification* for details.

The register is located in V_{DDCORE}.

If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPARE	MPUP1RSTF	MPUP0RSTF	CSTDBYRSTF	STDBYRSTF	Res.	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
rs	rs	rs	rs	rs		rs	rs	rs	rs		rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SPARE**: Spare bits
Set and reset by software. Reserved for future use.

Bit 14 **MPUP1RSTF**: MPU processor 1 reset flag
Set by the BOOTROM code when a MPU processor 1 reset occurred.
0: No MPU processor 1 reset occurred (default after nreset reset)
1: MPU processor 1 reset occurred

Bit 13 **MPUP0RSTF**: MPU processor 0 reset flag
Set by the BOOTROM code when a MPU processor 0 reset occurred.
0: No MPU processor 0 reset occurred (default after nreset reset)
1: MPU processor 0 reset occurred

Bit 12 **CSTDBYRSTF**: MPU CStandby reset flag
Set by the BOOTROM code when the MPU exits from CStandby.
0: MPU has not been in CStandby mode
1: MPU has been in CStandby mode

Bit 11 **STDBYRSTF**: System Standby reset flag
Set by the BOOTROM code when exiting from system Standby.
0: System has not been in Standby mode
1: System has been in Standby mode

Bit 10 Reserved, must be kept at reset value.

Bit 9 **IWDG2RSTF**: IWDG2 reset flag
Set by the BOOTROM code when a IWDG2 reset occurred.
0: No IWDG2 reset occurred
1: An IWDG2 reset occurred

Bit 8 **IWDG1RSTF**: IWDG1 reset flag
Set by the BOOTROM code when a IWDG1 reset occurred.
0: No IWDG1 reset occurred
1: An IWDG1 reset occurred

Bit 7 **MCSYSRSTF**: MCU System reset flag
Set by the BOOTROM code when a MCU system reset occurred.
0: No system reset generated by the MCU occurred
1: A system reset generated by the MCU occurred

Bit 6 **MPSYSRSTF**: MPU System reset flag
Set by the BOOTROM code when a MPU system reset occurred.
0: No system reset generated by the MPU occurred
1: A system reset generated by the MPU occurred

Bit 5 Reserved, must be kept at reset value.

Bit 4 **VCORERSTF**: V_{DDCORE} reset flag
Set by the BOOTROM code when a reset occurred because V_{DDCORE} is lower than the expected value.
0: V_{DDCORE} is not the origin of the reset
1: V_{DDCORE} is the origin of the reset

Bit 3 **HCSSRSTF**: HSE CSS reset flag
Set by the BOOTROM code when a failure is detected on HSE.
0: No HSE CSS reset occurred
1: A HSE CSS reset occurred

Bit 2 **PADRSTF**: NRST reset flag
Set by the BOOTROM code when a PAD reset occurred.
0: No PAD reset occurred
1: A PAD reset occurred

Bit 1 **BORRSTF**: BOR reset flag
Set by the BOOTROM code when a BOR reset occurred.
0: No BOR reset occurred
1: A BOR reset occurred

Bit 0 **PORRSTF**: POR/PDR reset flag
Set by the BOOTROM code when a POR/PDR reset occurred.
0: No POR/PDR reset occurred
1: A POR/PDR reset occurred

10.7.218 RCC IWDG clock freeze set register (RCC_MP_IWDGFZSETR)

Address offset: 0x40C

Reset value: 0x0000 0000

This register is used by the BOOTROM in order to freeze the IWDGs clocks. After a system reset or Standby reset (nreset), or a CStandby reset (cstby_rst) the MPU is allowed to write it once.Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' sets the corresponding bit to '1'. The MCU cannot modify this register. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FZ_IWDG2**: Freeze the IWDG2 clock

Set once by the BOOTROM software after a system reset or Standby reset, or a CStandby reset, in order to freeze the IWDG2.

0: Writing '0' has no effect, reading '0' means that the IWDG2 clock is not frozen (default after reset)

1: Writing '1' freeze the IWDG2 clock, reading '1' means that the IWDG2 clock is frozen

Bit 0 **FZ_IWDG1**: Freeze the IWDG1 clock

Set once by the BOOTROM software after a system reset or Standby reset, or a CStandby reset, in order to freeze the IWDG1.

0: Writing '0' has no effect, reading '0' means that the IWDG1 clock is not frozen (default after reset)

1: Writing '1' freeze the IWDG1 clock, reading '1' means that the IWDG1 clock is frozen

10.7.219 RCC IWDG clock freeze clear register (RCC_MP_IWDGFZCLRR)

Address offset: 0x410

Reset value: 0x0000 0000

This register is used by the BOOTROM in order to unfreeze the IWDGs clocks. Writing '0' has no effect, reading will return the effective values of the corresponding bits. Writing a '1' clears the corresponding bit to '0'. The MCU cannot modify this register. If TZEN = '1', this register can only be modified in secure mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FZ_IWDG2**: Unfreeze the IWDG2 clock

Cleared by the BOOTROM software, in order to unfreeze the IWDG2.

0: Writing '0' has no effect, reading '0' means that the IWDG2 clock is not frozen (default after reset)

1: Writing '1' unfreeze the IWDG2 clock, reading '1' means that the IWDG2 clock is frozen

Bit 0 **FZ_IWDG1**: Unfreeze the IWDG1 clock

Cleared by the BOOTROM software, in order to unfreeze the IWDG1.

0: Writing '0' has no effect, reading '0' means that the IWDG1 clock is not frozen (default after reset)

1: Writing '1' unfreeze the IWDG1 clock, reading '1' means that the IWDG1 clock is frozen

10.7.220 RCC version register (RCC_VERR)

Address offset: 0xFF4

Reset value: 0x0000 0011

This register gives the IP version

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major Revision of the IP

Bits 3:0 **MINREV[3:0]**: Minor Revision of the IP

10.7.221 RCC ID register (RCC_IDR)

Address offset: 0xFF8

Reset value: 0x0000 0001

This register gives the unique identifier of the RCC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier of the RCC

10.7.222 RCC size ID register (RCC_SIDR)

Address offset: 0xFFC

Reset value: 0xA3C5 DD04

This register gives the decoding space, which is for the RCC of 4 kB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Decoding space is 4 kbytes

10.8 RCC register map

Table 75. RCC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**RCC_TZCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCKPROT	TZEN
	Reset value																															1	1
0x004 - 0x008	Reserved	Res.																															
0x00C	**RCC_ OCENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSON	HSEBYP	HSEKERON	HSEON	DIGBYP	Res.	CSIKERON	CSION	Res.	Res.	HSIKERON	HSION
	Reset value																					0	0	0	0	0		0	0			0	1
0x010	**RCC_ OCENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEBYP	HSEKERON	HSEON	DIGBYP	Res.	CSIKERON	CSION	Res.	Res.	HSIKERON	HSION
	Reset value																						0	0	0	0		0	0			0	1
0x014	Res.																																
0x018	**RCC_HSICFGR**	Res.	Res.	Res.	Res.	HSICAL[11:0]												Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	HSIDIV[1:0]	
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0							0	0
0x01C	**RCC_CSICFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSICAL[7:0]								Res.	Res.	Res.	CSITRIM[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0				1	0	0	0	0								
0x020	**RCC_ MPCKSELR**	MPUSRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUSRC[1:0]	
	Reset value	1																														0	0
0x024	**RCC_ ASSCKSELR**	AXISSRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXISSRC[2:0]		
	Reset value	1																													0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	**RCC_ RCK12SELR**	PLL12SRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL12SRC[1:0]	
	Reset value	1																														0	0
0x02C	**RCC_MPCKDIVR**	MPUDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUDIV[2:0]		
	Reset value	1																													0	0	1
0x030	**RCC_AXIDIVR**	AXIDIVRD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIDIV[2:0]		
	Reset value	1																													0	0	0
0x034 - 0x038	Res.																																
0x03C	**RCC_APB4DIVR**	APB4DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB4DIV[2:0]		
	Reset value	1																													0	0	0
0x040	**RCC_APB5DIVR**	APB5DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB5DIV[2:0]		
	Reset value	1																													0	0	0
0x044	**RCC_RTCDIVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCDIV[5:0]					
	Reset value																											0	0	0	0	0	0
0x048	**RCC_ MSSCKSELR**	MCUSSRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCUSSRC[2:0]	
	Reset value	1																														0	0
0x04C - 0x07C	Reserved	Res.																															

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	**RCC_PLL1CR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVREN	DIVQEN	DIVPEN	Res	SSCG_CTRL	PLL1RDY	PLLON
	Reset value																										0	0	0		0	0	0
0x084	**RCC_ PLL1CFGR1**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVM1[5:0]						Res	Res	Res	Res	Res	Res	Res	DIVN[8:0]								
	Reset value											0	0	0	0	0	1								0	0	0	1	1	0	0	0	1
0x088	**RCC_ PLL1CFGR2**	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVR[6:0]							Res	DIVQ[6:0]							Res	DIVP[6:0]						
	Reset value										0	0	0	0	0	0	1		0	0	0	0	0	0	1		0	0	0	0	0	0	0
0x08C	**RCC_ PLL1FRACR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FRACLE	FRACV[12:0]													Res	Res	Res
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x090	**RCC_PLL1CSGR**	Res	INC_STEP[14:0]															SSCG_MODE	RPDFN_DIS	TPDFN_DIS	MOD_PER[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	**RCC_PLL2CR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVREN	DIVQEN	DIVPEN	Res	SSCG_CTRL	PLL2RDY	PLLON
	Reset value																										0	0	0		0	0	0
0x098	**RCC_ PLL2CFGR1**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVM2[5:0]						Res	Res	Res	Res	Res	Res	Res	DIVN[8:0]								
	Reset value											0	0	0	0	0	1								0	0	1	1	0	0	0	1	1
0x09C	**RCC_ PLL2CFGR2**	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVR[6:0]							Res	DIVQ[6:0]							Res	DIVP[6:0]						
	Reset value										0	0	0	0	0	0	1		0	0	0	0	0	0	1		0	0	0	0	0	0	1
0x0A0	**RCC_ PLL2FRACR**	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FRACLE	FRACV[12:0]													Res	Res	Res
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0A4	**RCC_PLL2CSGR**	Res	INC_STEP[14:0]															SSCG_MODE	RPDFN_DIS	TPDFN_DIS	MOD_PER[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A8 - 0x0BC	Reserved	Res.																															
0x0C0	**RCC_ I2C46CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C46SRC[2:0]		
	Reset value																														0	0	0
0x0C4	**RCC_ SPI6CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI6SRC[2:0]		
	Reset value																														0	0	0
0x0C8	**RCC_ UART1CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART1SRC[2:0]		
	Reset value																														0	0	0
0x0CC	**RCC_ RNG1CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1SRC[1:0]	
	Reset value																															0	0
0x0D0	**RCC_ CPERCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKPERSRC[1:0]	
	Reset value																															0	0
0x0D4	**RCC_ STGENCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSRC[1:0]	
	Reset value																															0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D8	**RCC_DDRITFCR**	GSKP_DUR[3:0]				DFILP_WIDTH[2:0]			GSKPCTRL	GSKPMOD	DDRCKMOD[2:0]			DPHYCTLRST	DPHYRST	DPHYAPBRST	DDRCORERST	DDRCAXIRST	DDRCAPBRST	KERDCG_DLY[2:0]			DDRPHYCAPBLPEN	DDRPHYCAPBEN	AXIDCGEN	DDRCAPBLPEN	DDRCAPBEN	DDRPHYCLPEN	DDRPHYCEN	DDRC2LPEN	DDRC2EN	DDRC1LPEN	DDRC1EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1	0	1	0
0x0DC - 0x0FC	Reserved	Res.																															
0x100	**RCC_MP_BOOTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPU_BEN	MCU_BEN
	Reset value																															0	0
0x104	**RCC_MP_SREQSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P1	STPREQ_P0
	Reset value																															0	0
0x108	**RCC_MP_SREQCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPREQ_P1	STPREQ_P0
	Reset value																															0	0
0x10C	**RCC_MP_GCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT_MCU
	Reset value																																0
0x110	**RCC_MP_APRSTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTTO[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDCTLEN
	Reset value																		1	1	1	1	1	1	1								0
0x114	**RCC_MP_APRSTSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTTOV[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x118 - 0x13C	Reserved	Res.																															

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x140	**RCC_BDCR**	VSWRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCCKEN	Res.	Res.	RTCSRC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	LSECSSD	LSECSSON	Res.	Res.	LSEDRV[1:0]		Res.	LSERDY	LSEBYP	LSEON
	Reset value	0											0			0	0							0	0			1	0		0	0	0
0x144	**RCC_RDLSICR**	SPARE[4:0]					EADLY[2:0]			Res.	Res.	Res.	MRD[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSIRDY	LSION
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0															0	0
0x148 - 0x17C	Reserved	Res.																															
0x180	**RCC_ APB4RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPHYRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCRST
	Reset value																0								0								0
0x184	**RCC_ APB4RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPHYRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCRST
	Reset value																0								0								0
0x188	**RCC_ APB5RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1RST	I2C6RST	I2C4RST	Res.	SPI6RST
	Reset value												0																0	0	0		0
0x18C	**RCC_ APB5RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USART1RST	I2C6RST	I2C4RST	Res.	SPI6RST
	Reset value												0																0	0	0		0
0x190	**RCC_ AHB5RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1RST	HASH1RST	CRYP1RST	Res.	Res.	Res.	GPIOZRST
	Reset value																0										0	0	0				0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x194	**RCC_ AHB5RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG1RST	HASH1RST	CRYP1RST	Res.	Res.	Res.	GPIOZRST
	Reset value																0										0	0	0				0
0x198	**RCC_ AHB6RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHRST	Res.	Res.	Res.	CRC1RST	Res.	Res.	SDMMC2RST	SDMMC1RST	Res.	QSPIRST	Res.	FMCRST	Res.	ETHMACRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0				0			0	0		0		0		0										
0x19C	**RCC_ AHB6RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHRST	Res.	Res.	Res.	CRC1RST	Res.	Res.	SDMMC2RST	SDMMC1RST	Res.	QSPIRST	Res.	FMCRST	Res.	ETHMACRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0				0			0	0		0		0		0										
0x1A0	**RCC_ TZAHB6RST SETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMARST
	Reset value																																0
0x1A4	**RCC_ TZAHB6RST CLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMARST
	Reset value																																0
0x1A8 - 0x1FC	Reserved	Res.																															
0x200	**RCC_MP_ APB4ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN	IWDG2APBEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
	Reset value												0				0	0							0								0
0x204	**RCC_MP_ APB4ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN	IWDG2APBEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
	Reset value												0				0	0							0								0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x208	**RCC_MP_ APB5ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN	IWDG1APBEN	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
	Reset value												0				0	0		0	0	0			0				0	0	0		0
0x20C	**RCC_MP_ APB5ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN	IWDG1APBEN	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
	Reset value												0				0	0		0	0	0			0				0	0	0		0
0x210	**RCC_MP_ AHB5ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
	Reset value																1								0		0	0	0				0
0x214	**RCC_MP_ AHB5ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXIMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
	Reset value																1								0		0	0	0				0
0x218	**RCC_MP_ AHB6ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN	Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	Reset value								0				0			0	0		0		0		0	0	0	0							0
0x21C	**RCC_MP_ AHB6ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN	Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	Reset value								0				0			0	0		0		0		0	0	0	0							0
0x220	**RCC_MP_ TZAHB6ENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	Reset value																																0
0x224	**RCC_MP_ TZAHB6ENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	Reset value																																0
0x228 - 0x27C	Reserved	Res.																															

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x280	**RCC_MC_ APB4ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
	Reset value												0				0								0								0
0x284	**RCC_MC_ APB4ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROEN	Res.	Res.	Res.	USBPHYEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCEN
	Reset value												0				0								0								0
0x288	**RCC_MC_ APB5ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN	Res.	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
	Reset value												0				0			0	0	0			0				0	0	0		0
0x28C	**RCC_MC_ APB5ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENEN	Res.	Res.	Res.	BSECEN	Res.	Res.	TZPCEN	TZC2EN	TZC1EN	Res.	Res.	RTCAPBEN	Res.	Res.	Res.	USART1EN	I2C6EN	I2C4EN	Res.	SPI6EN
	Reset value												0				0			0	0	0			0				0	0	0		0
0x290	**RCC_MC_ AHB5ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
	Reset value																								0		0	0	0				0
0x294	**RCC_MC_ AHB5ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMEN	Res.	RNG1EN	HASH1EN	CRYP1EN	Res.	Res.	Res.	GPIOZEN
	Reset value																								0		0	0	0				0
0x298	**RCC_MC_ AHB6ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN	Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	Reset value								0				0			0	0		0		0		0	0	0	0							0
0x29C	**RCC_MC_ AHB6ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHEN	Res.	Res.	Res.	CRC1EN	Res.	Res.	SDMMC2EN	SDMMC1EN	Res.	QSPIEN	Res.	FMCEN	Res.	ETHMACEN	ETHRXEN	ETHTXEN	ETHCKEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMAEN
	Reset value								0				0			0	0		0		0		0	0	0	0							0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2A0 - 0x2FC	Reserved	Res.																															
0x300	**RCC_MP_ APB4LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN	IWDG2APBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
	Reset value											0	1				1	1							1								1
0x304	**RCC_MP_ APB4LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN	IWDG2APBLPEN	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
	Reset value											0	1				1	1							1								1
0x308	**RCC_MP_ APB5LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN	IWDG1APBLPEN	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
	Reset value											0	1				1	1		1	1	1			1				1	1	1		1
0x30C	**RCC_MP_ APB5LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN	IWDG1APBLPEN	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
	Reset value											0	1				1	1		1	1	1			1				1	1	1		1
0x310	**RCC_MP_ AHB5LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
	Reset value																								1		1	1	1				1
0x314	**RCC_MP_ AHB5LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
	Reset value																								1		1	1	1				1

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x318	**RCC_MP_ AHB6LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN	Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	Reset value								1				1			1	1		1		1	0	1	1	1	1							1
0x31C	**RCC_MP_ AHB6LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN	Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	Reset value								1				1			1	1		1		1	0	1	1	1	1							1
0x320	**RCC_MP_ TZAHB6LPEN SETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	Reset value																																1
0x324	**RCC_MP_ TZAHB6LPEN CLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	Reset value																																1
0x328 - 0x37C	Reserved	Res.																															
0x380	**RCC_MC_ APB4LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
	Reset value											0	1				1								1								1
0x384	**RCC_MC_ APB4LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENROSTPEN	STGENROLPEN	Res.	Res.	Res.	USBPHYLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDRPERFMLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTDCLPEN
	Reset value											0	1				1								1								1
0x388	**RCC_MC_ APB5LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN	Res.	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
	Reset value											0	1				1			1	1	1			1				1	1	1		1

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38C	**RCC_MC_ APB5LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STGENSTPEN	STGENLPEN	Res.	Res.	Res.	BSECLPEN	Res.	Res.	TZPCLPEN	TZC2LPEN	TZC1LPEN	Res.	Res.	RTCAPBLPEN	Res.	Res.	Res.	USART1LPEN	I2C6LPEN	I2C4LPEN	Res.	SPI6LPEN
	Reset value											0	1				1			1	1	1			1				1	1	1		1
0x390	**RCC_MC_ AHB5LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
	Reset value																								1		1	1	1				1
0x394	**RCC_MC_ AHB5LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPSRAMLPEN	Res.	RNG1LPEN	HASH1LPEN	CRYP1LPEN	Res.	Res.	Res.	GPIOZLPEN
	Reset value																								1		1	1	1				1
0x398	**RCC_MC_ AHB6LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN	Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	Reset value								1				1			1	1		1		1	0	1	1	1	1							1
0x39C	**RCC_MC_ AHB6LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBHLPEN	Res.	Res.	Res.	CRC1LPEN	Res.	Res.	SDMMC2LPEN	SDMMC1LPEN	Res.	QSPILPEN	Res.	FMCLPEN	ETHSTPEN	ETHMACLPEN	ETHRXLPEN	ETHTXLPEN	ETHCKLPEN	Res.	Res.	Res.	Res.	Res.	Res.	MDMALPEN
	Reset value								1				1			1	1		1		1	0	1	1	1	1							1
0x3A0 - 0x3FC	Reserved	Res.																															
0x400	**RCC_BR_ RSTSCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUP1RSTF	MPUP0RSTF	Res.	Res.	Res.	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value																		0	0				0	0	0	0		1	0	1	0	1
0x404	**RCC_MP_ GRSTCSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPUP1RST	MPUP0RST	Res.	Res.	MCURST	MPSYSRST
	Reset value																											0	0			0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x408	RCC_MP_RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPARE	MPUP1RSTF	MPUP0RSTF	CSTDBYRSTF	STDBYRSTF	Res.	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value																	0	0	0	0	0		0	0	0	0		0	0	0	0	0
0x40C	RCC_MP_IWDGFZSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
	Reset value																															0	0
0x410	RCC_MP_IWDGFZCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FZ_IWDG2	FZ_IWDG1
	Reset value																															0	0
0x414	RCC_MP_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE	Res.	Res.	Res.	Res.	PLL4DYIE	PLL3DYIE	PLL2DYIE	PLL1DYIE	Res.	Res.	Res.	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value												0				0					0	0	0	0				0	0	0	0	0
0x418	RCC_MP_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF	Res.	Res.	Res.	Res.	PLL4DYF	PLL3DYF	PLL2DYF	PLL1DYF	Res.	Res.	Res.	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value												0				0					0	0	0	0				0	0	0	0	0
0x41C	RCC_PWRLPDLYCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCTMPSKP	Res.	Res.	PWRLP_DLY[21:0]																					
	Reset value								0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x420	RCC_MP_RSTSSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPARE	MPUP1RSTF	MPUP0RSTF	CSTDBYRSTF	STBBYRSTF	Res.	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	Res.	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value																	0	0	0	0	0		0	0	0	0		0	0	0	0	0
0x424 - 0x7FC	Reserved	Res.																															
0x800	RCC_MCO1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO1ON	Res.	Res.	Res.	Res.	MCO1DIV[3:0]				Res.	MCO1SEL[2:0]		
	Reset value																				0					0	0	0	0		0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804	RCC_ MCO2CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCO2ON	Res.	Res.	Res.	Res.	MCO2DIV[3:0]				Res.	MCO2SEL[2:0]		
	Reset value																				0					0	0	0	0		0	0	0
0x808	RCC_ OCRDYR	Res.	Res.	Res.	Res.	Res.	Res.	CKREST	AXICKRDY	MPUCKRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSERDY	Res.	Res.	Res.	CSIRDY	Res.	HSIDIVRDY	Res.	HSIRDY
	Reset value							0	0	0															0				0		0		0
0x80C	RCC_ DBGCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGRST	Res.	Res.	TRACECKEN	DBGCKEN	Res.	Res.	Res.	Res.	Res.	TRACEDIV[2:0]		
	Reset value																				0			0	0						0	0	1
0x810 - 0x81C	Reserved	Res.																															
0x820	RCC_ RCK3SELR	PLL3SRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3SRC[1:0]	
	Reset value	1																														0	0
0x824	RCC_ RCK4SELR	PLL4SRCRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4SRC[1:0]	
	Reset value	1																														0	0
0x828	RCC_ TIMG1PRER	TIMG1PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG1PRE
	Reset value	1																															0
0x82C	RCC_ TIMG2PRER	TIMG2PRERDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMG2PRE
	Reset value	1																															0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x830	**RCC_MCUDIVR**	MCUDIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCUDIV[3:0]			
	Reset value	1																												0	0	0	0
0x834	**RCC_APB1DIVR**	APB1DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB1DIV[2:0]		
	Reset value	1																													0	0	0
0x838	**RCC_APB2DIVR**	APB2DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB2DIV[2:0]		
	Reset value	1																													0	0	0
0x83C	**RCC_APB3DIVR**	APB3DIVRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIV[2:0]		
	Reset value	1																													0	0	0
0x840 - 0x87C	Reserved	Res.																															
0x880	**RCC_PLL3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVREN	DIVQEN	DIVPEN	Res.	Res.	PLL3RDY	PLLON
	Reset value																										0	0	0			0	0
0x884	**RCC_PLL3CFGR1**	Res.	Res.	Res.	Res.	Res.	Res.	IFRGE[1:0]		Res.	Res.	DIVM3[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVN[8:0]								
	Reset value							0	0			0	0	0	0	0	1								0	0	0	1	1	0	0	0	1
0x888	**RCC_PLL3CFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIVR[6:0]							Res.	DIVQ[6:0]							Res.	DIVP[6:0]						
	Reset value										0	0	0	0	0	0	1		0	0	0	0	0	0	1		0	0	0	0	0	0	1
0x88C	**RCC_PLL3FRACR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRACLE	FRACV[12:0]													Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x890	Res.																																

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x894	RCC_PLL4CR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVREN	DIVQEN	DIVPEN	Res	Res	PLL4RDY	PLLON
	Reset value																										0	0	0			0	0
0x898	RCC_ PLL4CFGR1	Res	Res	Res	Res	Res	Res	IFRGE[1:0]		Res	Res	DIVM4[5:0]						Res	Res	Res	Res	Res	Res	Res	DIVN[8:0]								
	Reset value							0	0			0	0	0	0	0	1								0	0	0	1	1	0	0	0	1
0x89C	RCC_ PLL4CFGR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	DIVR[6:0]							Res	DIVQ[6:0]							Res	DIVP[6:0]						
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x8A0	RCC_ PLL4FRACR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FRACLE	FRACV[12:0]													Res	Res	Res
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x8A4 - 0x8BC	Reserved	Res.																															
0x8C0	RCC_ I2C12CKSELR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	I2C12SRC[2:0]		
	Reset value																														0	0	0
0x8C4	RCC_ I2C35CKSELR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	I2C35SRC[2:0]		
	Reset value																														0	0	0
0x8C8	RCC_ SAI1CKSELR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SAI1SRC[2:0]		
	Reset value																														0	0	0
0x8CC	RCC_ SAI2CKSELR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SAI2SRC[2:0]		
	Reset value																														0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8D0	**RCC_ SAI3CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI3SRC[2:0]		
	Reset value																														0	0	0
0x8D4	**RCC_ SAI4CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI4SRC[2:0]		
	Reset value																														0	0	0
0x8D8	**RCC_ SPI2S1CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI1SRC[2:0]		
	Reset value																														0	0	0
0x8DC	**RCC_ SPI2S23CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI23SRC[2:0]		
	Reset value																														0	0	0
0x8E0	**RCC_ SPI45CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI45SRC[2:0]		
	Reset value																														0	0	0
0x8E4	**RCC_ UART6CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART6SRC[2:0]		
	Reset value																														0	0	0
0x8E8	**RCC_ UART24CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART24SRC[2:0]		
	Reset value																														0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8EC	RCC_UART35CKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART35SRC[2:0]		
	Reset value																														0	0	0
0x8F0	RCC_UART78CKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UART78SRC[2:0]		
	Reset value																														0	0	0
0x8F4	RCC_SDMMC12CKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC12SRC[2:0]		
	Reset value																														0	1	1
0x8F8	RCC_SDMMC3CKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3SRC[2:0]		
	Reset value																														0	0	0
0x8FC	RCC_ETHCKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETHPTPDIV[3:0]				Res.	Res.	ETHSRC[1:0]	
	Reset value																									0	0	0	0			0	0
0x900	RCC_QSPICKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPISRC[1:0]	
	Reset value																															0	0
0x904	RCC_FMCCKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSRC[1:0]	
	Reset value																															0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x908	Reserved	Res.																															
0x90C	**RCC_ FDCANCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANSRC[1:0]	
	Reset value																															0	0
0x910	Reserved	Res.																															
0x914	**RCC_ SPDIFCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPDIFSRC[1:0]	
	Reset value																															0	0
0x918	**RCC_ CECCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CECSRC[1:0]	
	Reset value																															0	0
0x91C	**RCC_ USBCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOSRC	Res.	Res.	USBPHYSRC[1:0]	
	Reset value																												0			0	0
0x920	**RCC_ RNG2CKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG2SRC[1:0]	
	Reset value																															0	0
0x924	Reserved	Res.																															
0x928	**RCC_ ADCCKSELR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADCSRC[1:0]	
	Reset value																															0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x92C	RCC_ LPTIM45CK SELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM45SRC[2:0]		
	Reset value																														0	0	0
0x930	RCC_ LPTIM23CK SELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM23SRC[2:0]		
	Reset value																														0	0	0
0x934	RCC_ LPTIM1CKSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM1SRC[2:0]		
	Reset value																														0	0	0
0x938 - 0x97C	Reserved	Res.																															
0x980	RCC_ APB1RSTSETR	MDIOSRST	Res.	DAC12RST	Res.	CECRST	SPDIFRST	Res.	I2C5RST	I2C3RST	I2C2RST	I2C1RST	Res.	UART8RST	UART7RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0		0		0	0		0	0	0	0		0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0
0x984	RCC_ APB1RSTCLRR	MDIOSRST	Res.	DAC12RST	Res.	CECRST	SPDIFRST	Res.	I2C5RST	I2C3RST	I2C2RST	I2C1RST	Res.	UART8RST	UART7RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0		0		0	0		0	0	0	0		0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0
0x988	RCC_ APB2RSTSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	DFSDMRST	Res.	SAI3RST	SAI2RST	SAI1RST	Res.	Res.	USART6RST	Res.	Res.	SPI5RST	SPI4RST	SPI1RST	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	TIM8RST	TIM1RST
	Reset value								0				0		0	0	0			0			0	0	0				0	0	0	0	0
0x98C	RCC_ APB2RSTCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	DFSDMRST	Res.	SAI3RST	SAI2RST	SAI1RST	Res.	Res.	USART6RST	Res.	Res.	SPI5RST	SPI4RST	SPI1RST	Res.	Res.	Res.	TIM17RST	TIM16RST	TIM15RST	TIM8RST	TIM1RST
	Reset value								0				0		0	0	0			0			0	0	0				0	0	0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x990	**RCC_ APB3RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRST	Res.	Res.	VREFRST	Res.	SYSCFGRST	Res.	Res.	SAI4RST	Res.	Res.	Res.	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST
	Reset value																0			0		0			0					0	0	0	0
0x994	**RCC_ APB3RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRST	Res.	Res.	VREFRST	Res.	SYSCFGRST	Res.	Res.	SAI4RST	Res.	Res.	Res.	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST
	Reset value																0			0		0			0					0	0	0	0
0x998	**RCC_ AHB2RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBORST	Res.	Res.	ADC12RST		Res.	DMAMUXRST	DMA2RST	DMA1RST
	Reset value																0								0			0			0	0	0
0x99C	**RCC_ AHB2RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBORST	Res.	Res.	ADC12RST		Res.	DMAMUXRST	DMA2RST	DMA1RST
	Reset value																0								0			0			0	0	0
0x9A0	**RCC_ AHB3RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCRST	HSEMRST	Res.	Res.	Res.	CRC2RST	RNG2RST	HASH2RST	CRYP2RST	Res.	Res.	Res.	DCMIRST
	Reset value																				0	0				0	0	0	0				0
0x9A4	**RCC_ AHB3RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCRST	HSEMRST	Res.	Res.	Res.	CRC2RST	RNG2RST	HASH2RST	CRYP2RST	Res.	Res.	Res.	DCMIRST
	Reset value																				0	0				0	0	0	0				0
0x9A8	**RCC_ AHB4RSTSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x9AC	**RCC_ AHB4RSTCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x9B0 - 0x9FC	Reserved	Res.																															

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA00	RCC_ MP_APB1EN SETR	MDIOSEN	Res.	DAC12EN	Res.	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0		0		0	0		0	0	0	0		0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0
0xA04	RCC_MP_ APB1ENCLRR	MDIOSEN	Res.	DAC12EN	Res.	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0		0		0	0		0	0	0	0		0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0
0xA08	RCC_MP_ APB2ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
	Reset value								0			0	0		0	0	0			0			0	0	0				0	0	0	0	0
0xA0C	RCC_MP_ APB2ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
	Reset value								0			0	0		0	0	0			0			0	0	0				0	0	0	0	0
0xA10	RCC_MP_ APB3ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN	Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
	Reset value												0				0			0		0			0					0	0	0	0
0xA14	RCC_MP_ APB3ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN	Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
	Reset value												0				0			0		0			0					0	0	0	0
0xA18	RCC_MP_ AHB2ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
	Reset value																0								0			0			0	0	0
0xA1C	RCC_MP_ AHB2ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
	Reset value																0								0			0			0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA20	**RCC_MP_ AHB3ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
	Reset value																				0	0				0	0	0	0				0
0xA24	**RCC_MP_ AHB3ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
	Reset value																				0	0				0	0	0	0				0
0xA28	**RCC_MP_ AHB4ENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0xA2C	**RCC_MP_ AHB4ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0xA30 - 0xA34	Reserved	Res.																															
0xA38	**RCC_MP_ MLAHBENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
	Reset value																												1				
0xA3C	**RCC_MP_ MLAHBENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
	Reset value																												1				
0xA40 - 0xA7C	Reserved	Res.																															
0xA80	**RCC_MC_ APB1ENSETR**	MDIOSEN	Res.	DAC12EN	WWDG1EN	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0		0	0	0	0		0	0	0	0		0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0
0xA84	**RCC_MC_ APB1ENCLRR**	MDIOSEN	Res.	DAC12EN	Res.	CECEN	SPDIFEN	Res.	I2C5EN	I2C3EN	I2C2EN	I2C1EN	Res.	UART8EN	UART7EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0		0		0	0		0	0	0	0		0	0	0	0	0	0		0	0		0	0	0	0	0	0	0	0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA88	RCC_MC_ APB2ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
	Reset value								0			0	0		0	0	0			0			0	0	0				0	0	0	0	0
0xA8C	RCC_MC_ APB2ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	ADFSDMEN	DFSDMEN	Res.	SAI3EN	SAI2EN	SAI1EN	Res.	Res.	USART6EN	Res.	Res.	SPI5EN	SPI4EN	SPI1EN	Res.	Res.	Res.	TIM17EN	TIM16EN	TIM15EN	TIM8EN	TIM1EN
	Reset value								0			0	0		0	0	0			0			0	0	0				0	0	0	0	0
0xA90	RCC_MC_ APB3ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN	Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
	Reset value												0				0			0		0			0					0	0	0	0
0xA94	RCC_MC_ APB3ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPEN	Res.	Res.	Res.	DTSEN	Res.	Res.	VREFEN	Res.	SYSCFGEN	Res.	Res.	SAI4EN	Res.	Res.	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN
	Reset value												0				0			0		0			0					0	0	0	0
0xA98	RCC_MC_ AHB2ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
	Reset value																0								0			0			0	0	0
0xA9C	RCC_MC_ AHB2ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOEN	Res.	Res.	ADC12EN	Res.	Res.	DMAMUXEN	DMA2EN	DMA1EN
	Reset value																0								0			0			0	0	0
0xAA0	RCC_MC_ AHB3ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
	Reset value																				0	0				0	0	0	0				0
0xAA4	RCC_MC_ AHB3ENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCEN	HSEMEN	Res.	Res.	Res.	CRC2EN	RNG2EN	HASH2EN	CRYP2EN	Res.	Res.	Res.	DCMIEN
	Reset value																				0	0				0	0	0	0				0
0xAA8	RCC_MC_ AHB4ENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xAAC	**RCC_MC_ AHB4ENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0xAB0	**RCC_MC_ AXIMENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMEN
	Reset value																																0
0xAB4	**RCC_MC_ AXIMENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMEN
	Reset value																																0
0xAB8	**RCC_MC_ MLAHBENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
	Reset value																												1				
0xABC	**RCC_MC_ MLAHBENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMEN	Res.	Res.	Res.	Res.
	Reset value																												1				
0xAC0 - 0xAFC	Reserved	Res.																															
0xB00	**RCC_MP_ APB1LPENSETR**	MDIOSLPEN	Res.	DAC12LPEN	Res.	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1		1		1	1		1	1	1	1		1	1	1	1	1	1		1	1		1	1	1	1	1	1	1	1	1	1
0xB04	**RCC_MP_ APB1LPENCLRR**	MDIOSLPEN	Res.	DAC12LPEN	Res.	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1		1		1	1		1	1	1	1		1	1	1	1	1	1		1	1		1	1	1	1	1	1	1	1	1	1

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB08	**RCC_MP_APB2LPENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN	Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
	Reset value								1			1	1		1	1	1			1			1	1	1				1	1	1	1	1
0xB0C	**RCC_MP_APB2LPENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN	Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
	Reset value								1			1	1		1	1	1			1			1	1	1				1	1	1	1	1
0xB10	**RCC_MP_APB3LPENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
	Reset value																1			1		1			1					1	1	1	1
0xB14	**RCC_MP_APB3LPENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
	Reset value																1			1		1			1					1	1	1	1
0xB18	**RCC_MP_AHB2LPENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
	Reset value																1								1			1			1	1	1
0xB1C	**RCC_MP_AHB2LPENCLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
	Reset value																1								1			1			1	1	1
0xB20	**RCC_MP_AHB3LPENSETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
	Reset value																				1	1				1	1	1	1				1

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB24	**RCC_MP_ AHB3LPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
	Reset value																				1	1				1	1	1	1				1
0xB28	**RCC_MP_ AHB4LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value																						1	1	1	1	1	1	1	1	1	1	1
0xB2 C	**RCC_MP_ AHB4LPEN CLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value																						1	1	1	1	1	1	1	1	1	1	1
0xB30	**RCC_MP_ AXIMLPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
	Reset value																																1
0xB34	**RCC_MP_ AXIMLPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
	Reset value																																1
0xB38	**RCC_MP_ MLAHBLPEN SETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
	Reset value																												1		1	1	1
0xB3 C	**RCC_MP_ MLAHBLPENCL RR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
	Reset value																												1		1	1	1
0xB40 - 0xB7 C	Reserved	Res.																															

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB80	RCC_MC_APB1LPENSETR	MDIOSLPEN	Res.	DAC12LPEN	WWDG1LPEN	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1		1	1	1	1		1	1	1	1		1	1	1	1	1	1		1	1		1	1	1	1	1	1	1	1	1	1
0xB84	RCC_MC_APB1LPENCLRR	MDIOSLPEN	Res.	DAC12LPEN	WWDG1LPEN	CECLPEN	SPDIFLPEN	Res.	I2C5LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	Res.	UART8LPEN	UART7LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1		1	1	1	1		1	1	1	1		1	1	1	1	1	1		1	1		1	1	1	1	1	1	1	1	1	1
0xB88	RCC_MC_APB2LPENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN	Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
	Reset value								1			1	1		1	1	1			1			1	1	1				1	1	1	1	1
0xB8C	RCC_MC_APB2LPENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	ADFSDMLPEN	DFSDMLPEN	Res.	SAI3LPEN	SAI2LPEN	SAI1LPEN	Res.	Res.	USART6LPEN	Res.	Res.	SPI5LPEN	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM8LPEN	TIM1LPEN
	Reset value								1			1	1		1	1	1			1			1	1	1				1	1	1	1	1
0xB90	RCC_MC_APB3LPENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
	Reset value																1			1		1			1					1	1	1	1
0xB94	RCC_MC_APB3LPENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	Res.	Res.	VREFLPEN	Res.	SYSCFGLPEN	Res.	Res.	SAI4LPEN	Res.	Res.	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN
	Reset value																1			1		1			1					1	1	1	1
0xB98	RCC_MC_AHB2LPENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
	Reset value																1								1			1			1	1	1

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB9C	**RCC_MC_ AHB2LPEN CLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC3LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBOLPEN	Res.	Res.	ADC12LPEN	Res.	Res.	DMAMUXLPEN	DMA2LPEN	DMA1LPEN
	Reset value																1								1			1			1	1	1
0xBA0	**RCC_MC_ AHB3LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
	Reset value																				1	1				1	1	1	1				1
0xBA4	**RCC_MC_ AHB3LPEN CLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPCCLPEN	HSEMLPEN	Res.	Res.	Res.	CRC2LPEN	RNG2LPEN	HASH2LPEN	CRYP2LPEN	Res.	Res.	Res.	DCMILPEN
	Reset value																				1	1				1	1	1	1				1
0xBA8	**RCC_MC_ AHB4LPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value																						1	1	1	1	1	1	1	1	1	1	1
0xBAC	**RCC_MC_ AHB4LPEN CLRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value																						1	1	1	1	1	1	1	1	1	1	1
0xBB0	**RCC_MC_ AXIMLPENSET R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
	Reset value																																1
0xBB4	**RCC_MC_ AXIMLPENCLR R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSRAMLPEN
	Reset value																																1
0xBB8	**RCC_MC_ MLAHBLPEN SETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
	Reset value																												1		1	1	1

Table 75. RCC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBBC	RCC_MC_MLAHBLPENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RETRAMLPEN	Res.	SRAM34LPEN	SRAM2LPEN	SRAM1LPEN
	Reset value																												1		1	1	1
0xBC0 - 0xBFC	Reserved	Res.																															
0xC00	RCC_MC_RSTSCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1RSTF	IWDG2RSTF	IWDG1RSTF	MCSYSRSTF	MPSYSRSTF	MCURSTF	VCORERSTF	HCSSRSTF	PADRSTF	BORRSTF	PORRSTF
	Reset value																						0	0	0	0	0	0	1	0	1	0	1
0xC04 - 0xC10	Reserved	Res.																															
0xC14	RCC_MC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	LSECSSIE	Res.	Res.	Res.	Res.	PLL4DYIE	PLL3DYIE	PLL2DYIE	PLL1DYIE	Res.	Res.	Res.	CSIRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value												0				0					0	0	0	0				0	0	0	0	0
0xC18	RCC_MC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	LSECSSF	Res.	Res.	Res.	Res.	PLL4DYF	PLL3DYF	PLL2DYF	PLL1DYF	Res.	Res.	Res.	CSIRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
	Reset value												0				0					0	0	0	0				0	0	0	0	0
0xC1C - 0xFF0	Reserved	Res.																															
0xFF4	RCC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFF8	RCC_IDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0xFFC	RCC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	1	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

11 Hardware semaphore (HSEM)

11.1 Introduction

The hardware semaphore block provides 32 (32-bit) register based semaphores.

The semaphores can be used to ensure synchronization between different processes running between different cores. The HSEM provides a non-blocking mechanism to lock semaphores in an atomic way. The following functions are provided:

- Semaphore lock, in two ways:
  - 2-step lock: by writing COREID and PROCID to the semaphore, followed by a read check
  - 1-step lock: by reading the COREID from the semaphore
- Interrupt generation when a semaphore is unlocked
  - Each semaphore may generate an interrupt on one of the interrupt lines
- Semaphore clear protection
  - A semaphore is only unlocked when COREID and PROCID match
- Global semaphore clear per COREID

11.2 Main features

The HSEM includes the following features:

- 32 (32-bit) semaphores
- 8-bit PROCID
- 4-bit COREID
- One interrupt line per processor
- Lock indication

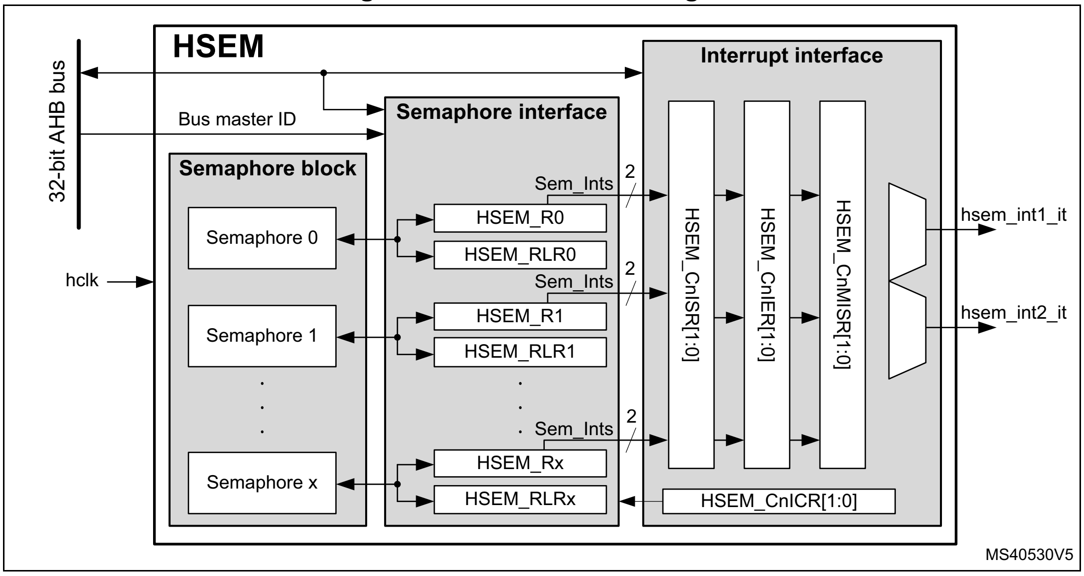
11.3 Functional description

11.3.1 HSEM block diagram

As shown in *Figure 107*, the HSEM is based on three sub-blocks:

- the semaphore block containing the semaphore status and IDs
- the semaphore interface block providing AHB access to the semaphore via the HSEM_Rx and HSEM_RLRx registers
- the interrupt interface block providing control for the interrupts via HSEM_CnISR, HSEM_CnIER, HSEM_CnMISR, and HSEM_CnICR registers.

Figure 107. HSEM block diagram



11.3.2 HSEM internal signals

Table 76. HSEM internal input/output signals

Signal name	Signal type	Description
AHB bus	Digital input/output	AHB register access bus
BusMasterID	Digital input	AHB bus master ID
hsem_intn_it	Digital output	Interrupt n line (n = 1 to 2)

11.3.3 HSEM lock procedures

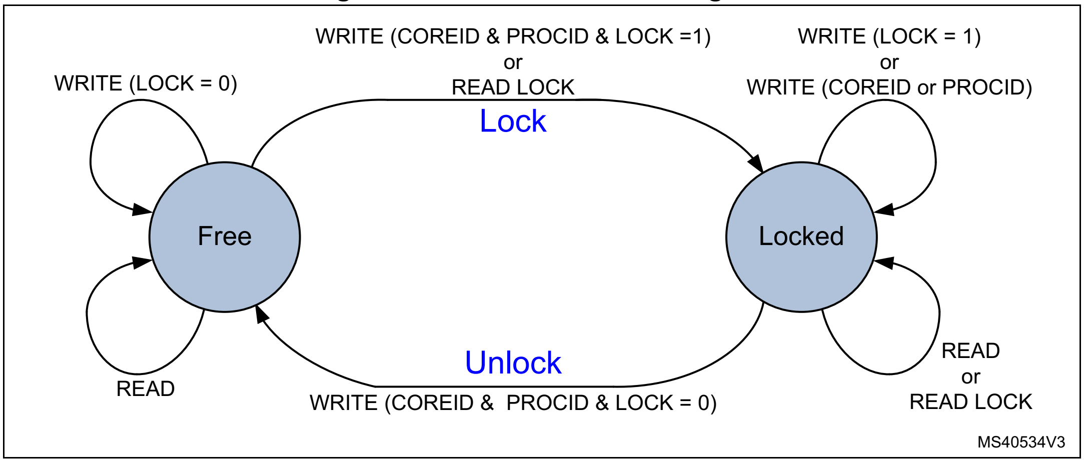
There are two lock procedures, namely 2-step (write) lock and 1-step (read) lock. The two procedures can be used concurrently.

The semaphore is free when its LOCK bit is 0. In this case, the COREID and PROCID are also 0. When the LOCK bit is 1, the semaphore is locked and the COREID indicates which AHB bus master ID has locked it. The PROCID indicates which process of that AHB bus master ID has locked the semaphore.

When write locking a semaphore, the written COREID must match the AHB bus master ID , and the PROCID is written by the AHB bus master software process taking the lock.

When read locking the semaphore, the COREID is taken from the AHB bus master ID, and the PROCID is forced to 0 by hardware. There is no PROCID available with read lock.

Figure 108. Procedure state diagram



2-step (write) lock procedure

The 2-step lock procedure consists in a write to lock the semaphore, followed by a read to check if the lock has been successful, carried out from the HSEM_Rx register.

- Write semaphore with PROCID and COREID, and LOCK = 1. The COREID data written by software must match the AHB bus master information, that is, a AHB bus master ID = 1 writes data COREID = 1.

  Lock is put in place when the semaphore is free at write time.
- Read-back the semaphore

  The software checks the lock status, if PROCID and COREID match the written data, then the lock is confirmed.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free.

A semaphore can be locked when the PROCID = 0.

Consecutive write attempts with LOCK = 1 to a locked semaphore are ignored.

1-step (read) lock procedure

The 1-step procedure consists in a read to lock and check the semaphore in a single step, carried out from the HSEM_RLRx register.

- Read lock semaphore with the AHB bus master COREID.
- If read COREID matches and PROCID = 0, then lock is put in place. If COREID matches and PROCID is not 0, this means that another process from the same COREID has locked the semaphore with a 2-step (write) procedure.
- Else retry (the semaphore has been locked by another process, AHB bus master ID).

A semaphore can only be locked when it is free. When read locking a free semaphore, PROCID is 0. Read locking a locked semaphore returns the COREID and PROCID that locked it. All read locks, including the first one that locks the semaphore, return the COREID that locks or locked the semaphore.

Note: The 1-step procedure must not be used when running multiple processes of the same AHB bus master ID. All processes using the same semaphore read the same status. When only one process locks the semaphore, each process of that AHB bus master ID reads the semaphore as locked by itself with the COREID.

11.3.4 HSEM write/read/read lock register address

For each semaphore, two AHB register addresses are provided, separated in two banks of 32-bit semaphore registers, spaced by a 0x80 address offset.

In the first register address bank the semaphore can be written (locked/unlocked) and read through the HSEM_Rx registers.

In the second register address bank the semaphore can be read (locked) through the HSEM_RLRx registers.

11.3.5 HSEM unlock procedures

Unlocking a semaphore is a protected process, to prevent accidental clearing by a AHB bus master ID or by a process not having the semaphore lock right. The procedure consists in writing to the semaphore HSEM_Rx register with the corresponding COREID and PROCID and LOCK = 0. When unlocked, the COREID, and the PROCID are all 0.

When unlocked, an interrupt may be generated to signal the event. To this end, the semaphore interrupt must be enabled.

The unlock procedure consists in a write to the semaphore HSEM_Rx register with matching COREID regardless on how the semaphore has been locked (1- or 2-step).

- Write semaphore with PROCID, COREID, and LOCK = 0
- If the written data matches the semaphore PROCID and COREID and the AHB bus master ID , the semaphore is unlocked and an interrupt may be generated when enabled, else write is ignored, semaphore remains locked and no interrupt is generated (the semaphore is locked by another process, AHB bus master ID or the written data does not match the AHB bus master signaling).

Note: Different processes of the same AHB bus master ID can write any PROCID value. Preventing other processes of the same AHB bus master ID from unlocking a semaphore must be ensured by software, handling the PROCID correctly.

11.3.6 HSEM COREID semaphore clear

All semaphores locked by a COREID can be unlocked at once by using the HSEM_CR register. Write COREID and correct KEY value in HSEM_CR. All locked semaphores with a matching COREID are unlocked, and may generate an interrupt when enabled.

Note: This procedure may be used in case of an incorrect functioning AHB bus master ID, where another AHB bus master can unlock the locked semaphores by writing the COREID of the incorrect functioning processor into the HSEM_CR register with the correct KEY value. This unlocks all locked semaphores with a matching COREID.

An interrupt may be generated for the unlocked semaphore(s). To this end, the semaphore interrupt must be enabled in the HSEM_CnIER registers.

11.3.7 HSEM interrupts

An interrupt line hsem_intn_it per processor allows each semaphore to generate an interrupt.

An interrupt line provides the following features per semaphore:

- interrupt enable
- interrupt clear
- interrupt status
- masked interrupt status

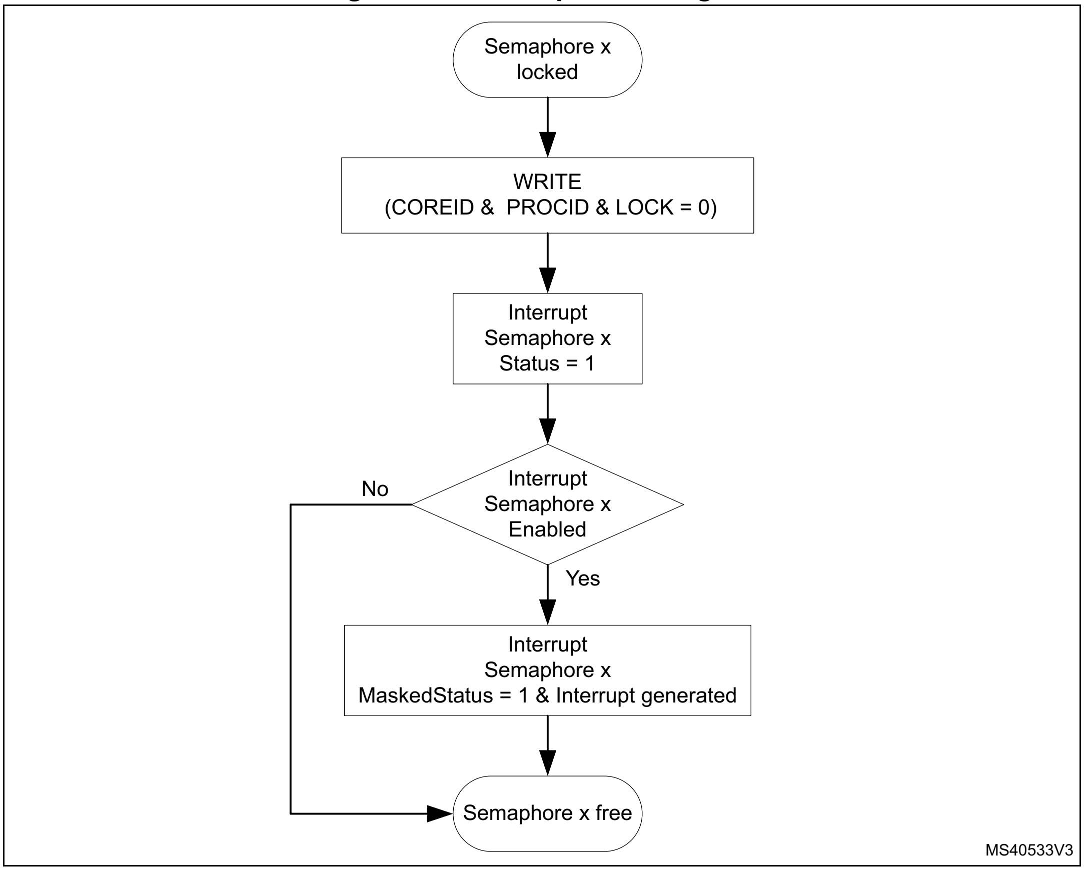
With the interrupt enable (HSEM_CnIER) the semaphores affecting the interrupt line can be enabled. Disabled (masked) semaphore interrupts do not set the masked interrupt status MISF for that semaphore, and do not generate an interrupt on the interrupt line.

The interrupt clear (HSEM_CnICR) clears the interrupt status ISF and masked interrupt status MISF of the associated semaphore for the interrupt line.

The interrupt status (HSEM_CnISR) mirrors the semaphore interrupt status ISF before the enable.

The masked interrupt status (HSEM_CnMISR) only mirrors the semaphore enabled interrupt status MISF on the interrupt line. All masked interrupt status MISF of the enabled semaphores need to be cleared to clear the interrupt line.

Figure 109. Interrupt state diagram



The procedure to get an interrupt when a semaphore becomes free is described hereafter.

Try to lock semaphore x

- If the semaphore lock is obtained, no interrupt is needed.
- If the semaphore lock fails:
- Clear pending semaphore x interrupt status for the interrupt line in HSEM_CnICR.
  Re-try to lock the semaphore x again:
  - If the semaphore lock is obtained, no interrupt is needed (semaphore has been freed between first try to lock and clear semaphore interrupt status).
  - If the semaphore lock fails, enable the semaphore x interrupt in HSEM_CnIER.

On semaphore x free interrupt, try to lock semaphore x

- If the semaphore lock is obtained:
  Disable the semaphore x interrupt in HSEM_CnIER.
  Clear pending semaphore x interrupt status in HSEM_CnICR.
- If the semaphore x lock fails:
  Clear pending semaphore x interrupt status in HSEM_CnICR.
  Try again to lock the semaphore x:
  - If the semaphore lock is obtained (semaphore has been freed between first try to lock and semaphore interrupt status clear), disable the semaphore interrupt in HSEM_CnIER.

– If the semaphore lock fails, wait for semaphore free interrupt.

Note: An interrupt does not lock the semaphore. After an interrupt, either the AHB bus master or the process must still perform the lock procedure to lock the semaphore.

It is possible to have multiple AHB bus masters informed by the semaphore free interrupts. Each AHB bus master gets its interrupt, and the first one to react locks the semaphore.

11.3.8 AHB bus master ID verification

The HSEM allows only authorized AHB bus master IDs to lock and unlock semaphores.

- The AHB bus master 2-step lock write access to the semaphore HSEM_Rx register is checked against the valid bus master IDs.
  – Accesses from unauthorized AHB bus master IDs are discarded and do not lock the semaphore.
- The AHB bus master 1-step lock read access from the semaphore HSEM_RLRx register is checked against the valid bus master IDs.
  – An unauthorized AHB bus master ID read from HSEM_RLRx returns all 0.
- The semaphore unlock write access to the HSEM_CR register is checked against the valid bus master IDs. Only the valid bus master IDs can write to the HSEM_CR register and unlock any of the COREID semaphores.
  – Accesses from unauthorized AHB bus master IDs are discarded and do not clear the COREID semaphores.

Table 77 details the relation between bus master/processor and COREID.

Table 77. Authorized AHB bus master IDs

Bus master 0 (processor1)	Bus master 1 (processor2)
COREID = 1	COREID = 2

Note: Accesses from unauthorized AHB bus master IDs to other registers are granted.

11.4 HSEM registers

Registers must be accessed using word format. Byte and half-word accesses are ignored and have no effect on the semaphores, they generate a bus error.

11.4.1 HSEM register semaphore x (HSEM_Rx)

Address offset: 0x000 + 0x4 * x (x = 0 to 31)

Reset value: 0x0000 0000

The HSEM_Rx must be used to perform a 2-step write lock, read back, and for unlocking a semaphore. Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	COREID[3:0]				PROCID[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK:** Lock indication

This bit can be written and read by software.

0: On write free semaphore (only when COREID and PROCID match), on read semaphore is free.

1: On write try to lock semaphore, on read semaphore is locked.

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** Semaphore COREID

Written by software

- When the semaphore is free and the LOCK bit is at the same time written to 1 and the COREID matches the AHB bus master ID.
- When the semaphore is unlocked (LOCK written to 0 and AHB bus master ID matched COREID, the COREID is cleared to 0.
- When the semaphore is unlocked (LOCK bit written to 0 or AHB bus master ID does not match COREID, the COREID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the COREID is not affected.
- An authorized read returns the stored COREID value.

Bits 7:0 **PROCID[7:0]:** Semaphore PROCID

Written by software

-When the semaphore is free and the LOCK is written to 1, and the COREID matches the AHB bus master ID, PROCID is set to the written data.
- When the semaphore is unlocked, LOCK written to 0 and AHB bus master ID matched COREID, the PROCID is cleared to 0.
- When the semaphore is unlocked, LOCK bit written to 0 and AHB bus master ID does not match COREID, the PROCID is not affected.
- Write when LOCK bit is already 1 (semaphore locked), the PROCID is not affected.
- An authorized read returns the stored PROCID value.

11.4.2 HSEM read lock register semaphore x (HSEM_RLRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 31)

Reset value: 0x0000 0000

Accesses the same physical bits as HSEM_Rx. The HSEM_RLRx must be used to perform a 1-step read lock. Only read accesses with authorized AHB bus master IDs are granted. Read accesses with unauthorized AHB bus master IDs are discarded and return 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	COREID[3:0]				PROCID[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **LOCK:** Lock indication

This bit is read only by software at this address.

- When the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and returns 1.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns 1 (the COREID and PROCID reflect the already locked semaphore information).

Bits 30:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** Semaphore COREID

This field is read only by software at this address.

On a read, when the semaphore is free, the hardware sets the COREID to the AHB bus master ID reading the semaphore. The COREID of the AHB bus master locking the semaphore is read.

On a read when the semaphore is locked, this field returns the COREID of the AHB bus master that has locked the semaphore.

Bits 7:0 **PROCID[7:0]:** Semaphore processor ID

This field is read only by software at this address.

- On a read when the semaphore is free:

A read with a valid AHB bus master ID locks the semaphore and hardware sets the PROCID to 0.

- When the semaphore is locked:

A read with a valid AHB bus master ID returns the PROCID of the AHB bus master that has locked the semaphore.

11.4.3 HSEM interrupt enable register (HSEM_CnIER)

Address offset: 0x100 + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ISE[31:0]:** Interrupt(n) semaphore x enable bit (x = 0 to 31)
This bit is read and written by software.
0: Interrupt(n) generation for semaphore x disabled (masked)
1: Interrupt(n) generation for semaphore x enabled (not masked)

11.4.4 HSEM interrupt clear register (HSEM_CnICR)

Address offset: 0x104 + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISC[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISC[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ISC[31:0]:** Interrupt(n) semaphore x clear bit (x = 0 to 31)
This bit is written by software, and is always read 0.
0: Interrupt(n) semaphore x status ISFx and masked status MISFx not affected.
1: Interrupt(n) semaphore x status ISFx and masked status MISFx cleared.

11.4.5 HSEM interrupt status register (HSEM_CnISR)

Address offset: 0x108 + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ISF[31:0]:** Interrupt semaphore x status bit before enable (mask) (x = 0 to 31)
This bit is set by hardware, and reset only by software. This bit is cleared by software writing the corresponding HSEM_CnICR bit.
0: Interrupt semaphore x status, no interrupt pending
1: Interrupt semaphore x status, interrupt pending

11.4.6 HSEM interrupt status register (HSEM_CnMISR)

Address offset: 0x10C + 0x010 * (n - 1), (n = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MISF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MISF[31:0]:** Masked interrupt(n) semaphore x status bit after enable (mask) (x = 0 to 31)
This bit is set by hardware and read only by software. This bit is cleared by software writing the corresponding HSEM_CnICR bit. This bit is read as 0 when semaphore x status is masked in HSEM_CnIER bit x.
0: interrupt(n) semaphore x status after masking not pending
1: interrupt(n) semaphore x status after masking pending

11.4.7 HSEM clear register (HSEM_CR)

Address offset: 0x140

Reset value: 0x0000 0000

Only write accesses with authorized AHB bus master IDs are granted. Write accesses with unauthorized AHB bus master IDs are discarded.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	COREID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				w	w	w	w								

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written by software and is always read 0.
If this key value does not match HSEM_KEYR.KEY, semaphores are not affected.
If this key value matches HSEM_KEYR.KEY, all semaphores matching the COREID are cleared to the free state.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **COREID[3:0]:** COREID of semaphores to be cleared
This field can be written by software and is always read 0.
This field indicates the COREID for which the semaphores are cleared when writing the HSEM_CR.

Bits 7:0 Reserved, must be kept at reset value.

11.4.8 HSEM clear semaphore key register (HSEM_KEYR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **KEY[15:0]:** Semaphore clear key
This field can be written and read by software.
Key value to match when clearing semaphores.

Bits 15:0 Reserved, must be kept at reset value.

11.4.9 HSEM hardware configuration register 2 (HSEM_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MASTERID4[3:0]				MASTERID3[3:0]				MASTERID2[3:0]				MASTERID1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **MASTERID4[3:0]:** Hardware configuration valid bus masters ID4

Bits 11:8 **MASTERID3[3:0]:** Hardware configuration valid bus masters ID3

Bits 7:4 **MASTERID2[3:0]:** Hardware configuration valid bus masters ID2

Bits 3:0 **MASTERID1[3:0]:** Hardware configuration valid bus masters ID1

11.4.10 HSEM hardware configuration register 1 (HSEM_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0000 0220

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	NBINT[3:0]				NBSEM[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **NBINT[3:0]**: Hardware configuration number of interrupts/supported number of master ID.

Bits 7:0 **NBSEM[7:0]**: Hardware configuration number of semaphores

11.4.11 HSEM version register (HSEM_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: HSEM major revision number

Bits 3:0 **MINREV[3:0]**: HSEM minor revision number

11.4.12 HSEM identification register (HSEM_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0072

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]**: HSEM identifier

11.4.13 HSEM size identification register (HSEM_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: HSEM size

11.4.14 HSEM register map

Table 78. HSEM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HSEM_R0	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x004	HSEM_R1	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x07C	HSEM_R31	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x080	HSEM_RLR0	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x084	HSEM_RLR1	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
⋮																																	
0x0FC	HSEM_RLR31	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREID [3:0]				PROCID[7:0]							
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x100	HSEM_C1IER	ISE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	HSEM_C1ICR	ISC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	HSEM_C1ISR	ISF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	HSEM_C1MISR	MISF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110	HSEM_C2IER	ISE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	HSEM_C2ICR	ISC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118	HSEM_C2ISR	ISF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 78. HSEM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x11C	HSEM_C2MISR	MISF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	HSEM_CR	KEY[15:0]																Res.	Res.	Res.	Res.	COREID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0								
0x144	HSEM_KEYR	KEY[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x3EC	HSEM_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTERID4				MASTERID3				MASTERID2				MASTERID1			
	Reset value																	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0x3F0	HSEM_HWCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBINT				NBSEM							
	Reset value																					0	0	1	0	0	0	1	0	0	0	0	0
0x3F4	HSEM_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV				MINREV			
	Reset value																									0	0	1	0	0	0	0	0
0x3F8	HSEM_IPIDR	IPID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0
0x3FC	HSEM_SIDR	SID																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

12 Inter-processor communication controller (IPCC)

12.1 Introduction

The inter-processor communication controller (IPCC) is used for communicating data between two processors.

The IPCC block provides a nonblocking signaling mechanism to post and retrieve communication data in an atomic way. It provides the signaling for twelve channels:

- six channels in the direction from processor 1 to processor 2
- six channels in the opposite direction

It is then possible to have two different communication types in each direction.

The IPCC communication data must be located in a common memory, which is not part of the IPCC block.

12.2 IPCC main features

- Status signaling for the twelve channels
  – Channel occupied/free flag, also used as lock
- Two interrupt lines per processor
  – One for RX channel occupied (communication data posted by sending processor)
  – One for TX channel free (communication data retrieved by receiving processor)
- Interrupt masking per channel
  – Channel occupied mask
  – Channel free mask
- Two channel operation modes
  – Simplex (each channel has its own communication data memory location)
  – Half duplex (a single channel associated to a bidirectional communication data information memory location)

12.3 IPCC functional description

The IPCC communication data is located in a common memory, which is not part of the IPCC block. The address location of the communication data must be known or located in a known common area that, as already stated, is not part of the IPCC block.

For each communication, the IPCC block provides a channel status flag CHnF.

- When 0, the channel status flag CHnF indicates that the associated IPCC channel is free (the receiving processor has retrieved communication data), and can be accessed by the sending processor.
- When 1, the channel status flag CHnF indicates that the associated IPCC channel is occupied (the sending processor has posted communication data) and can be accessed by the receiving processor.

The channel operation mode must be known to both processors. A common parameter can be used to indicate the channel transfer mode and must also be located in a known common area. This parameter is not available from the IPCC.

12.3.1 IPCC block diagram

The IPCC (see *Figure 110*) consists of the following subblocks:

- Status block, containing the channel status
- IPCC interface block, providing AHB access to the channel status registers
- Interrupt interface block, providing control for the interrupts

Figure 110. IPCC block diagram

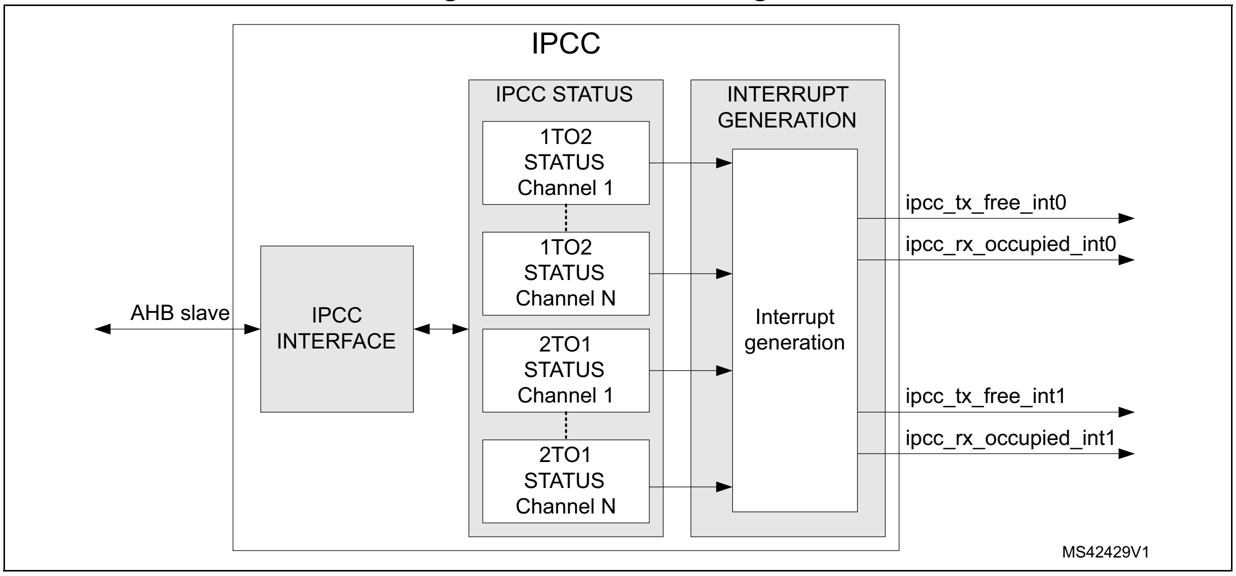


Table 79. IPCC interface signals

Signal		Description
Name	Type	
AHB slave	I/O	AHB register access bus
ipcc_tx_free_int1	O	TX free interrupt to processor 1
ipcc_rx_occupied_int1	O	RX occupied interrupt to processor 1
ipcc_tx_free_int2	O	TX free interrupt to processor 2
ipcc_rx_occupied_int2	O	RX occupied interrupt to processor 2

12.3.2 IPCC Simplex channel mode

In Simplex channel mode, a dedicated memory location (used to transfer data in a single direction) is assigned to the communication data. The associated channel N control bits (see *Table 80*) are used to manage the transfer from the sending to the receiving processor.

Table 80. Bits used for the communication

Processor	A	B
SEND A = 1 RECEIVE B = 2	IPCC_C1CR.TXFIE IPCC_C1MR.CHnFM IPCC_C1SCR.CHnS IPCC_C1TOC2SR.CHnF	IPCC_C2CR.RXOIE IPCC_C2MR.CHnOM IPCC_C2SCR.CHnC
SEND A = 2 RECEIVE B = 1	IPCC_C2CR.TXFIE IPCC_C2MR.CHnFM IPCC_C2SCR.CHnS IPCC_C2TOC1SR.CHnF	IPCC_C1CR.RXOIE IPCC_C1MR.CHnOM IPCC_C1SCR.CHnC

Once the sending processor has posted the communication data in the memory, it sets the channel status flag CHnF to occupied with CHnS.

Once the receiving processor has retrieved the communication data from the memory, it clears the channel status flag CHnF back to free with CHnC.

Figure 111. IPCC Simplex channel mode transfer timing

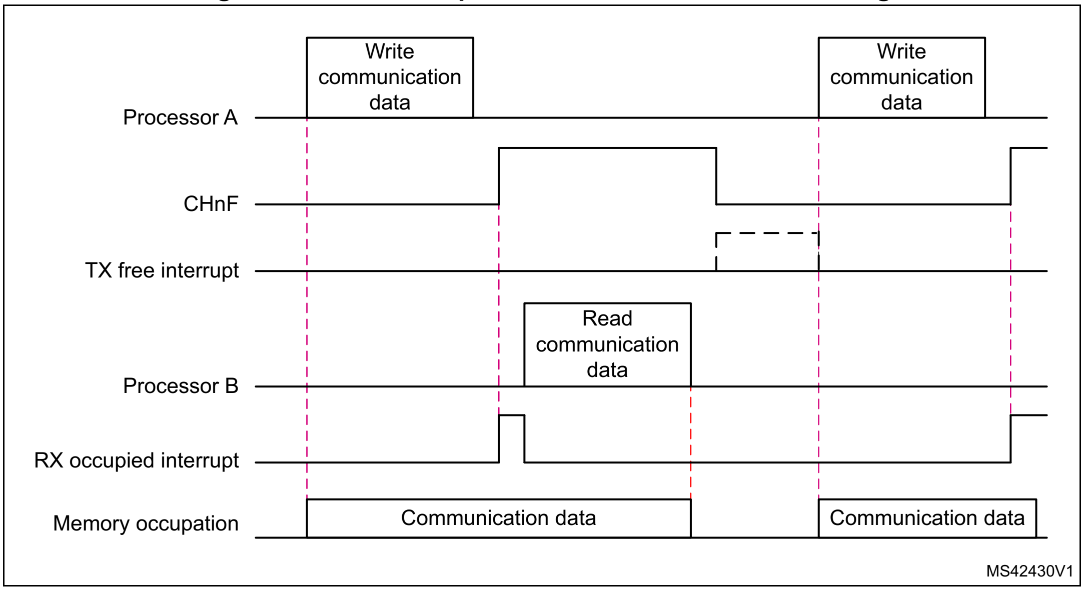
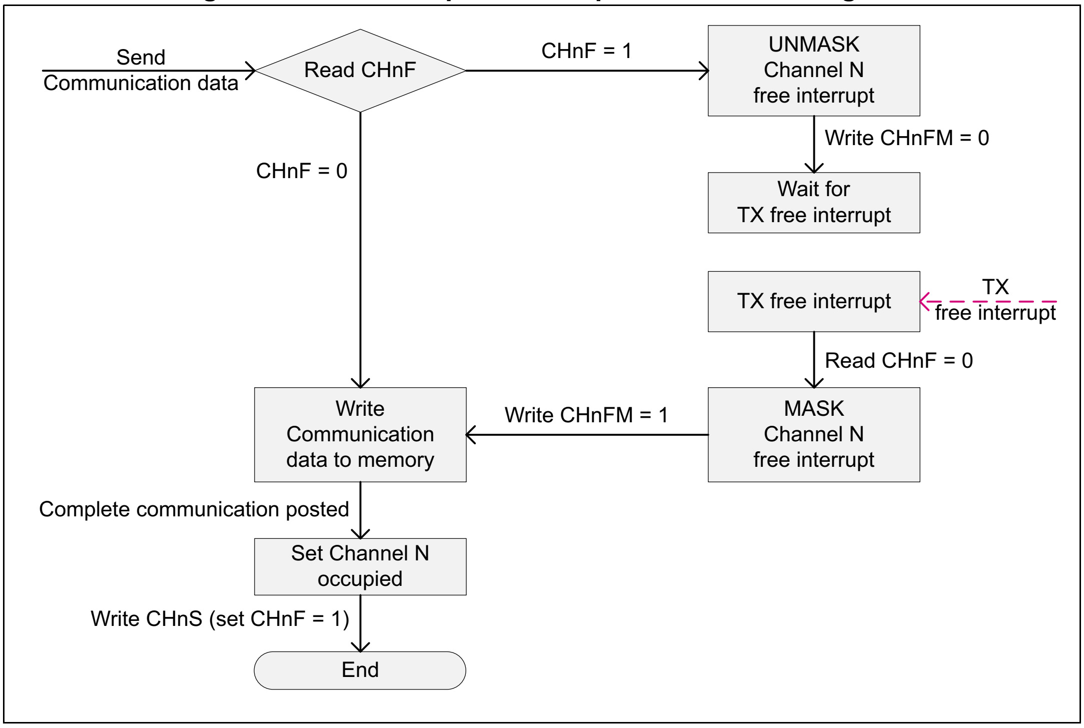


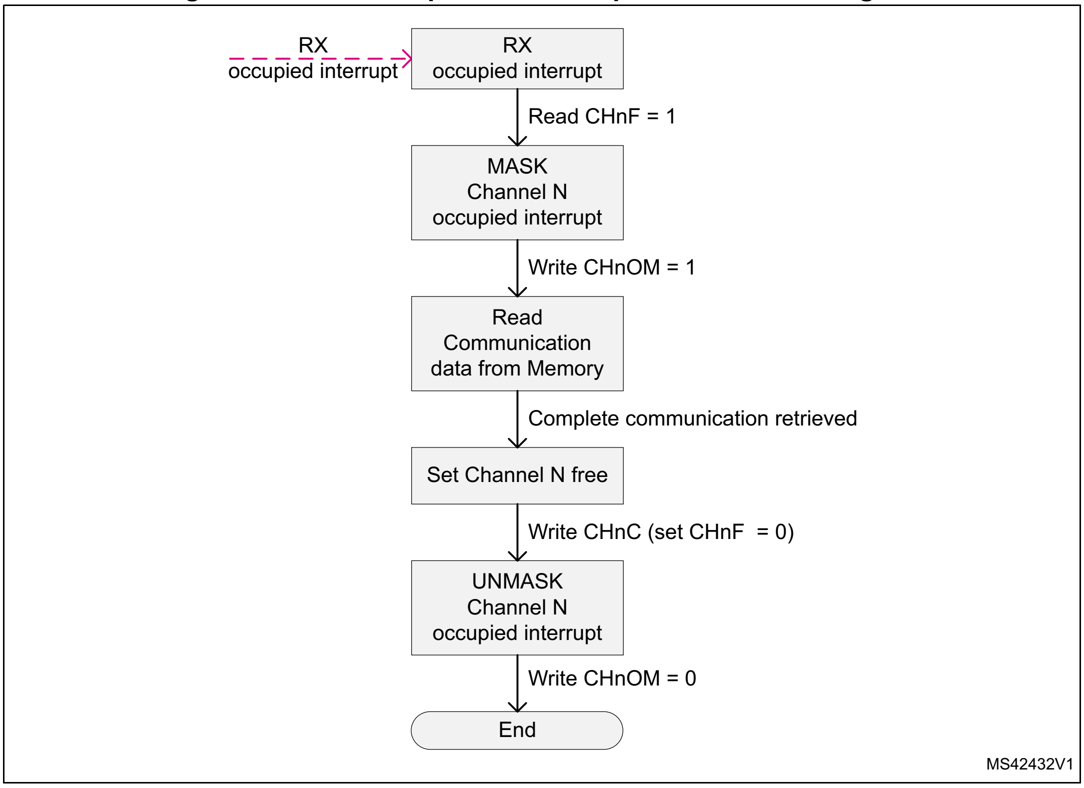
Figure 112. IPCC Simplex - Send procedure state diagram



To send communication data:

- The sending processor checks the channel status flag CHnF:
  - When CHnF = 0, the channel is free (last communication data retrieved by the receiving processor), and the new communication data can be written.
  - When CHnF = 1, the channel is occupied (last communication data not retrieved by the receiving processor), and the sending processor unmasks the channel free interrupt (CHnFM = 0).
  - On a TX free interrupt, the sending processor checks which channel became free and masks the channel free interrupt (CHnFM = 1). Then the new communication can take place.
- Once the complete communication data is posted, the channel status is set to occupied with CHnS. This gives memory access to the receiving processor and generates the RX occupied interrupt.

Figure 113. IPCC Simplex - Receive procedure state diagram



To receive a communication, the channel occupied interrupt is unmasked (CHnOM = 0):

- On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from memory.
- Once the complete communication data is retrieved, the channel status is cleared to free with CHnC. This gives memory access back to the sending processor and may generate the TX free interrupt.
- Once the channel status is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

12.3.3 IPCC Half-duplex channel mode

The Half-duplex channel mode is used when one processor sends a communication and the other processor sends a response to each communication (ping-pong).

In Half-duplex channel mode, a single dedicated memory location is assigned to communication data and response, and is used to transfer data in both directions. The sending processor channel status flag CHnF is assigned to the channel and used by both processors (see *Table 80*).

Once the processor A posts communication data into memory, it sets the processor A channel status flag CHnF to occupied with CHnS (giving memory access to processor B).

Once the processor B retrieves communication data from memory, it does not change the channel status flags. The memory access is kept by processor B for the response.

Once the processor B posts the response into memory, it clears the channel status flag CHnF to free with CHnC (giving memory access back to processor A).

Once the processor A retrieves the response from the memory, it does not change the channel status flags. The memory location access is kept by processor A for the next communication data.

Figure 114. IPCC Half-duplex channel mode transfer timing

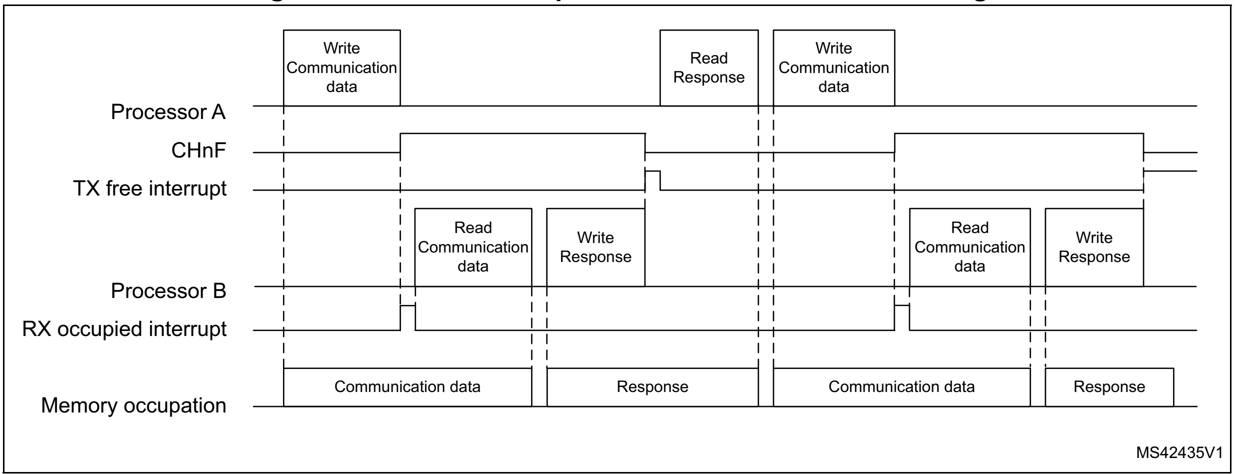
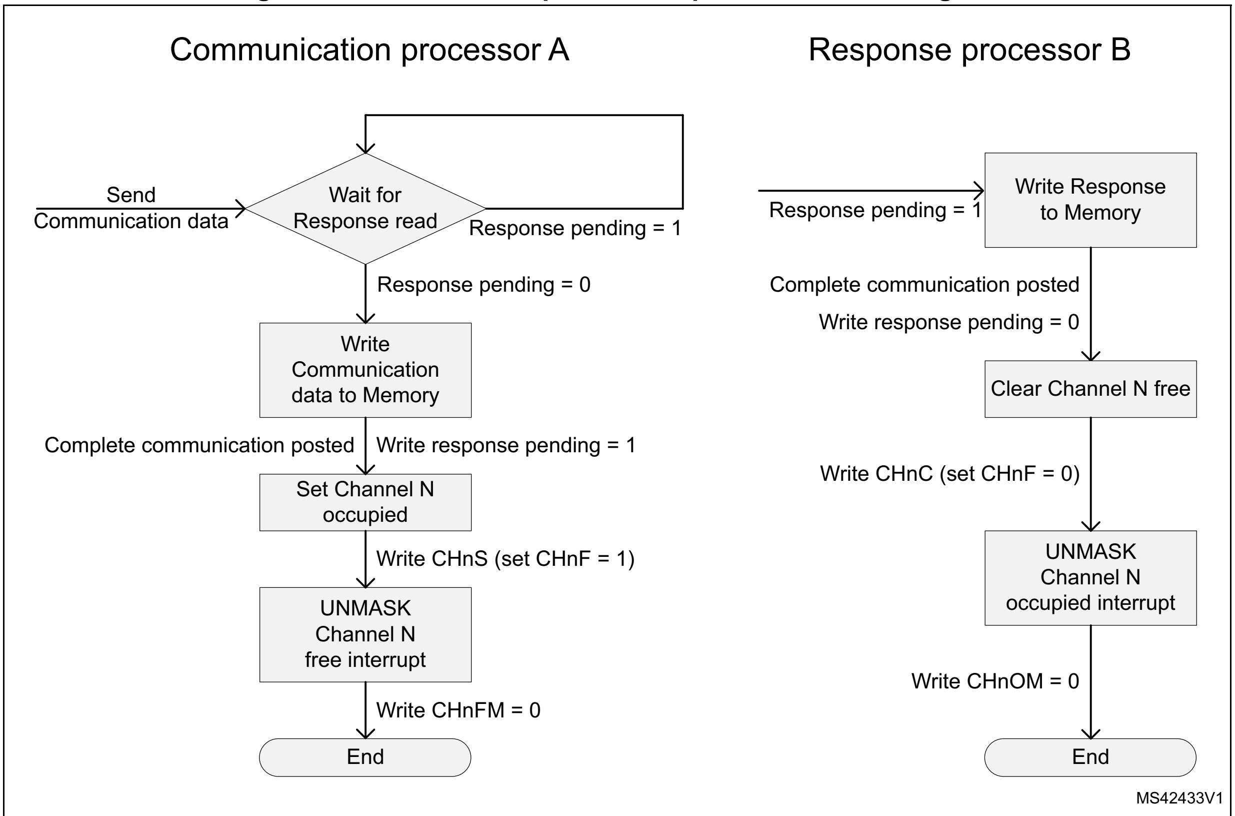


Figure 115. IPCC Half-duplex - Send procedure state diagram



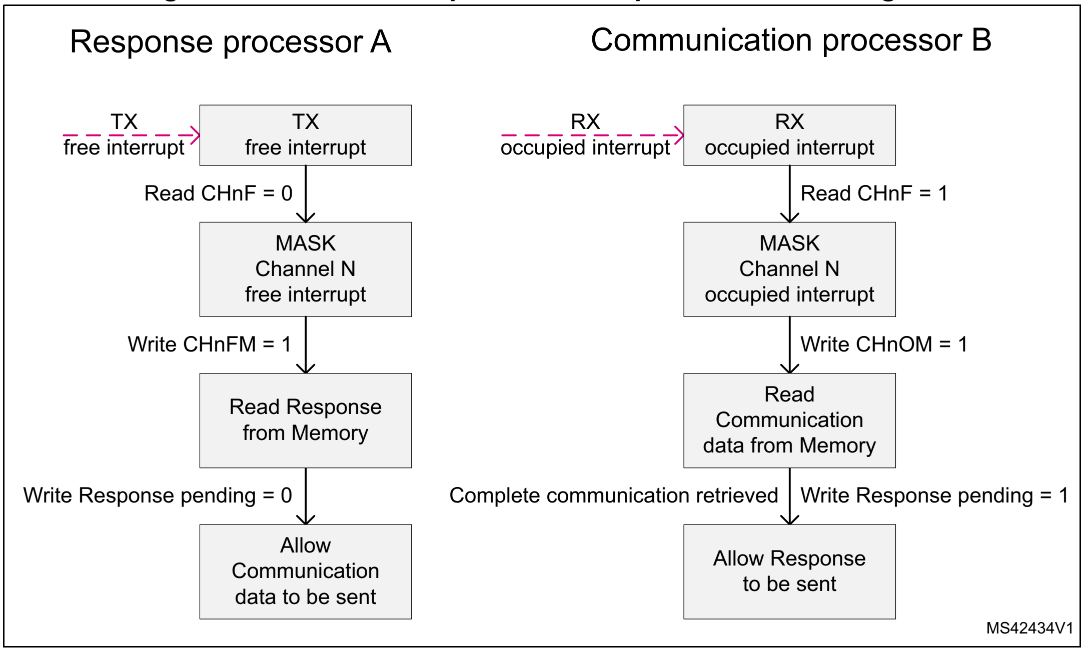
To send communication data:

- The sending processor waits for its response pending software variable to get 0.
  – Once the response pending software variable is 0 the communication data is posted.
- Once the complete communication data has been posted, the channel status flag CHnF is set to occupied with CHnS, and the response pending software variable is set to 1 (this gives memory access and generates the RX occupied interrupt to the receiving processor).
- Once the channel status flag CHnF is set, the channel free interrupt is unmasked (CHnFM = 0).

To send a response:

- The receiving processor waits for its response pending software variable to get 1.
  – Once the response pending software variable is 1, the response is posted.
- Once the complete response is posted, the channel status flag CHnF is cleared to free with CHnC, and the response pending software variable is set to 0 (this gives memory access and generates the TX free interrupt to the sending processor).
- Once the channel status flag CHnF is cleared, the channel occupied interrupt is unmasked (CHnOM = 0).

Figure 116. IPCC Half-duplex - Receive procedure state diagram



To receive communication data the channel occupied interrupt is unmasked (CHnOM = 0):

- On an RX occupied interrupt, the receiving processor checks which channel became occupied, masks the associated channel occupied interrupt (CHnOM) and reads the communication data from the memory.
- Once the complete communication data is retrieved, the response pending software variable is set. The channel status is not changed, access to the memory is kept to post the subsequent response.

To receive the response the channel free interrupt is unmasked (CHnFM = 0):

- On a TX free interrupt, the sending processor checks which channel became free, masks the associated channel free interrupt (CHnFM) and reads the response from the memory.
- Once the complete response is retrieved, the response pending software variable is cleared. The channel status is not changed, access to the memory is kept to post the subsequent communication data.

12.3.4 IPCC interrupts

There are four interrupt lines :

- two RX channel occupied interrupts, one for each processor:
  - Interrupt enable RXOIE per processor
  - Individual mask CHnOM per channel
- two TX channel free interrupts, one for each processor
  - Interrupt enable TXFIE per processor
  - Individual mask CHnFM per channel

The RX occupied interrupt is used by the receiving processor and indicates when an unmasked channel status indicates occupied (CHnF = 1).

The TX free interrupt is used by the sending processor, and indicates when an unmasked channel status indicates free (CHnF = 0).

A secure channel only generates a secure interrupt, and only in the case when the channel is secure unmasked and global secure enabled.

A non-secure channel only generates a non-secure interrupt, and only in the case when the channel is non-secure unmasked and global non-secure enabled.

12.4 IPCC registers

The peripheral registers must be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses are not permitted and do not generate a bus error.

12.4.1 IPCC processor 1 control register (IPCC_C1CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFIE
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TXFIE**: Processor 1 transmit channel free interrupt enable
Associated with IPCC_C1TOC2SR.
1: Enable an unmasked processor 1 transmit channel free to generate a TX free interrupt.
0: Processor 1 TX free interrupt disabled

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RXOIE**: Processor 1 receive channel occupied interrupt enable
Associated with IPCC_C2TOC1SR.
1: Enable an unmasked processor 1 receive channel occupied to generate an RX occupied interrupt.
0: Processor 1 RX occupied interrupt disabled

12.4.2 IPCC processor 1 mask register (IPCC_C1MR)

Address offset: 0x004

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6 FM	CH5 FM	CH4 FM	CH3 FM	CH2 FM	CH1 FM
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6 OM	CH5 OM	CH4 OM	CH3 OM	CH2 OM	CH1 OM
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnFM:** Processor 1 transmit channel n status set, (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnOM**: Processor 1 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

12.4.3 IPCC processor 1 status set clear register (IPCC_C1SCR)

Address offset: 0x008

Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnS:** Processor 1 transmit channel n status set (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Processor 1 transmit channel n status bit set.
0: No action.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnC**: Processor 1 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Processor 1 receive channel n status bit clear.
0: No action.

12.4.4 IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnF:** Processor 1 transmit to processor 2 receive channel n status flag before masking (n = 6 to 1).
1: Channel occupied, data can be read by the receiving processor 2.
Generates a channel RX occupied interrupt to processor 2, when unmasked.
0: Channel free, data can be written by the sending processor 1.
Generates a channel TX free interrupt to processor 1, when unmasked.

12.4.5 IPCC processor 2 control register (IPCC_C2CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFIE
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE
															rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TXFIE:** Processor 2 transmit channel free interrupt enable
Associated with IPCC_C2TOC1SR.
1: Enable an unmasked processor 2 transmit channel free to generate a TX free interrupt.
0: Processor 2 TX free interrupt disabled

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **RXOIE:** Processor 2 receive channel occupied interrupt enable
Associated with IPCC_C1TOC2SR.
1: Enable an unmasked processor 2 receive channel occupied to generate an RX occupied interrupt.
0: Processor 2 RX occupied interrupt disabled

12.4.6 IPCC processor 2 mask register (IPCC_C2MR)

Address offset: 0x014

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6 FM	CH5 FM	CH4 FM	CH3 FM	CH2 FM	CH1 FM
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6 OM	CH5 OM	CH4 OM	CH3 OM	CH2 OM	CH1 OM
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnFM:** Processor 2 transmit channel n free interrupt mask (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Transmit channel n free interrupt masked.
0: Transmit channel n free interrupt not masked.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnOM**: Processor 2 receive channel n occupied interrupt mask (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Receive channel n occupied interrupt masked.
0: Receive channel n occupied interrupt not masked.

12.4.7 IPCC processor 2 status set clear register (IPCC_C2SCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reading this register always returns 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S
										rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
										rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CHnS:** Processor 2 transmit channel n status set (n = 6 to 1).
Associated with IPCC_C2TOC1SR.CHnF
1: Processor 2 transmit channel n status bit set.
0: No action.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnC**: Processor 2 receive channel n status clear (n = 6 to 1).
Associated with IPCC_C1TOC2SR.CHnF
1: Processor 2 receive channel n status bit clear.
0: No action.

12.4.8 IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **CHnF:** Processor 2 transmit to processor 1 receive channel n status flag before masking (n = 6 to 1)
- 1: Channel occupied, data can be read by the receiving processor 1.
  Generates a channel RX occupied interrupt to processor 1, when unmasked.
- 0: Channel free, data can be written by the sending processor 2.
  Generates a channel TX free interrupt to processor 2, when unmasked.

12.4.9 IPCC hardware configuration register (IPCC_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CIDWIDTH[3:0]				OPTBITS[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	SECCTRL[3:0]				CHANNELS[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **CIDWIDTH[3:0]:** CID bit field width range 0 to 12.
- 0: No CID filtering
- 1..12: CID filtering with CID bit field with 1..12
- 13..15: Reserved

Bits 23:16 **OPTBITS[7:0]:** Reserved for future use

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **SECCTRL[3:0]:** Security options, range 1 to 16.
- 0: No security
- 1: Security
- 2: Security and privilege
- 3: Security, privilege and resource isolation.

Bits 7:0 **CHANNELS[7:0]:** Number of channels per CPU supported by the IP, range 1 to 16

12.4.10 IPCC version register (IPCC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** IPCC major revision number

Bits 3:0 **MINREV[3:0]:** IPCC minor revision number.

12.4.11 IPCC identification register (IPCC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0071

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]**: IPCC identification.

12.4.12 IPCC size identification register (IPCC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: IPCC size identification.

12.4.13 IPCC register map

Table 81. IPCC register map and reset values

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	IPCC_C1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE
	Reset value																0																0
0x0004	IPCC_C1MR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6FM	CH5FM	CH4FM	CH3FM	CH2FM	CH1FM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM
	Reset value											1	1	1	1	1	1											1	1	1	1	1	1
0x0008	IPCC_C1SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
	Reset value											0	0	0	0	0	0											0	0	0	0	0	0
0x000C	IPCC_C1TOC2SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
	Reset value																											0	0	0	0	0	0
0x0010	IPCC_C2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE
	Reset value																0																0
0x0014	IPCC_C2MR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6FM	CH5FM	CH4FM	CH3FM	CH2FM	CH1FM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM
	Reset value											1	1	1	1	1	1											1	1	1	1	1	1
0x0018	IPCC_C2SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6S	CH5S	CH4S	CH3S	CH2S	CH1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6C	CH5C	CH4C	CH3C	CH2C	CH1C
	Reset value											0	0	0	0	0	0											0	0	0	0	0	0
0x001C	IPCC_C2TOC1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F
	Reset value																											0	0	0	0	0	0
0x03F0	IPCC_HWCFGR	Res.	Res.	Res.	Res.	CIDWIDTH [3:0]				OPTBITS[7:0]								Res.	Res.	Res.	Res.	SECCTRL [3:0]				CHANNELS[7:0]							
	Reset value					0	0	1	1	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	1	1	0
0x03F4	IPCC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x03F8	IPCC_PIDR	IPID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1
0x03FC	IPCC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

13 General-purpose I/Os (GPIO)

13.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

13.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

13.3 GPIO functional description

Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words, half-words or bytes. The purpose of the GPIOx_BSRR register is to allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 117 and *Figure 118* show the basic structures of a standard and a 5-Volt tolerant I/O port bit, respectively. *Table 82* gives the possible port bit configurations.

Figure 117. Basic structure of an I/O port bit

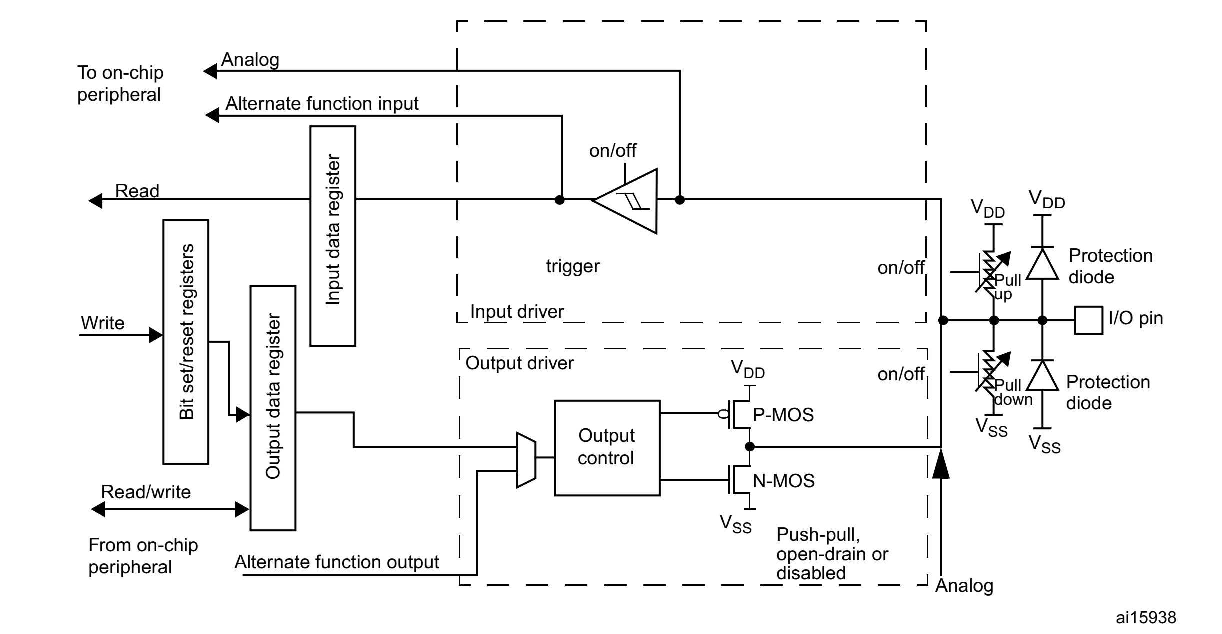
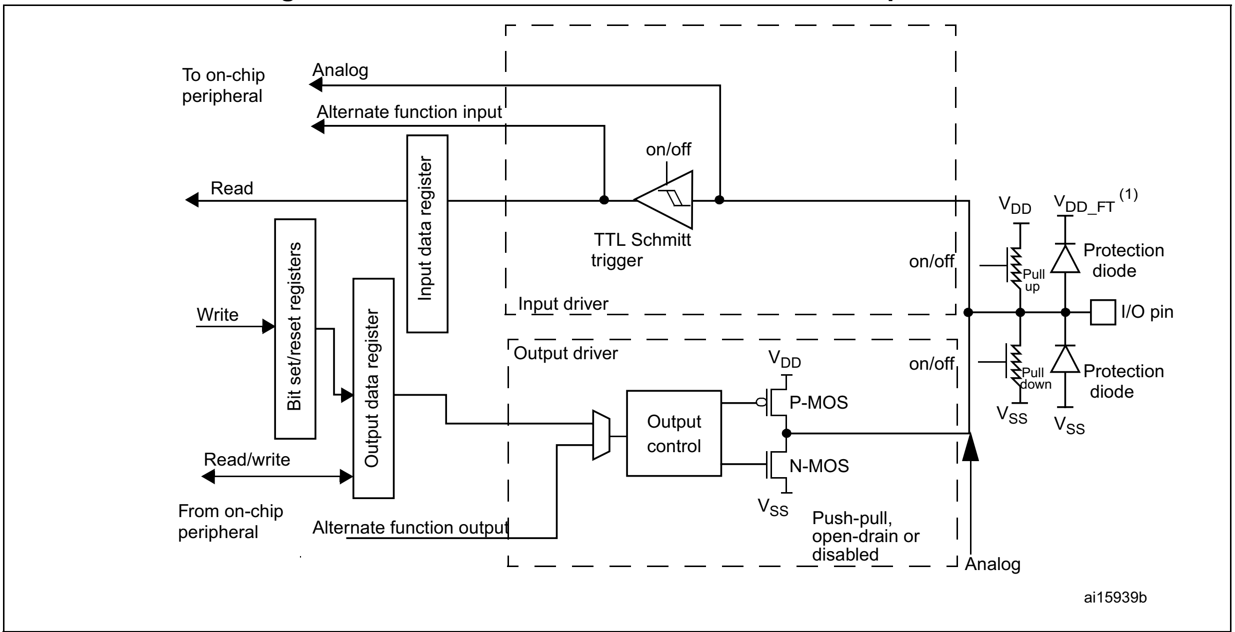


Figure 118. Basic structure of a 5-Volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Table 82. Port bit configuration table[(1)]

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0	Input/output	Analog, PD
	x	x	x	1	1	Reserved	

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

13.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in input floating mode.

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PUPDR register.

13.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to sixteen alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.
- Cortex-M4 with FPU EVENTOUT is mapped on AF15

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user has to proceed as follows:

- **System function:** MCOx pins have to be configured in alternate function mode.
- **GPIO:** configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function:**
  – Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  – Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER registers, respectively.
  – Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  – For the ADC and DAC, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and DAC registers.
  – For the additional functions like OTG_ID, OTG_FS_DP/DM, RTC_OUT, RTC_LSCO, TAMP_xxx, WKUPx and oscillators, configure the required function in the related OTG, RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers. For details about I/O control by the RTC, refer to *Section 48.3: RTC functional description*.
- EVENTOUT
  – Configure the I/O pin used to output the core EVENTOUT signal by connecting it to AF15.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

13.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

13.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

See *Section 13.4.5: GPIO port input data register (GPIOx_IDR) (x = A to K, Z)* and *Section 13.4.6: GPIO port output data register (GPIOx_ODR) (x = A to K, Z)* for the register descriptions.

13.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register which allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: it is possible to modify one or more bits in a single atomic AHB write access.

13.3.6 GPIO locking mechanism

It is possible to freeze the GPIO control registers by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write / read sequence has to be applied. When the right LOCK sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence has been applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence (refer to *Section 13.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to K, Z)*) can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 has to be set at the same time as the [15:0] bits.

For more details refer to LCKR register description in *Section 13.4.8: GPIO port configuration lock register (GPIOx_LCKR) (x = A to K, Z)*.

13.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin refer to the device datasheet.

13.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port must not be configured in analog mode.

Refer to *Section 23: Extended interrupt and event controller (EXTI).*

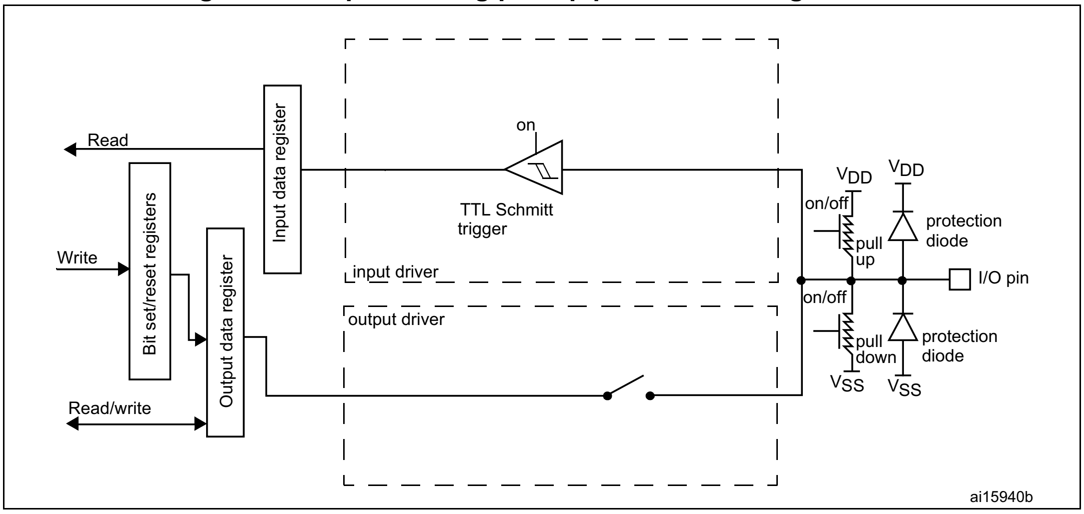
13.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register provides the I/O state

Figure 119 shows the input configuration of the I/O port bit.

Figure 119. Input floating/pull up/pull down configurations



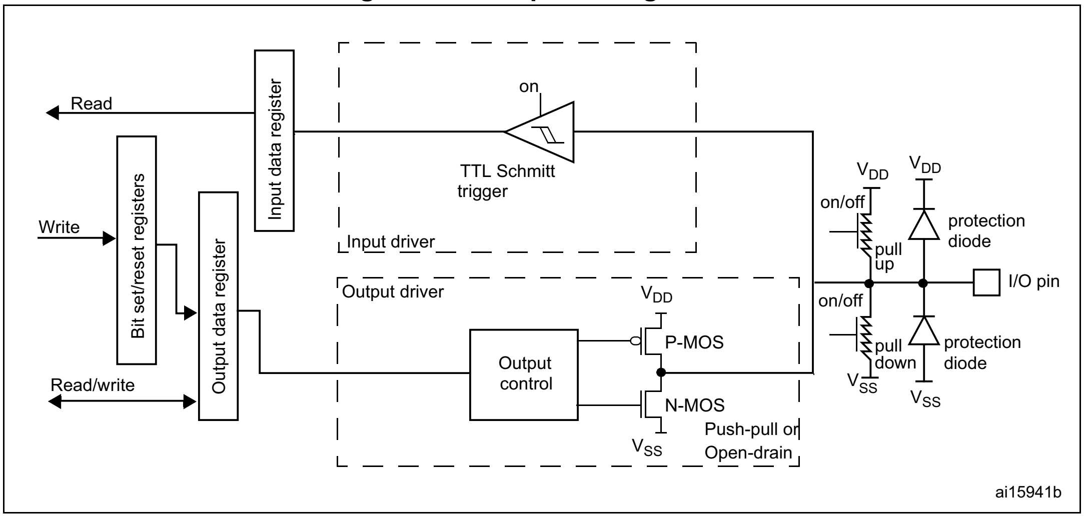
13.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open drain mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register leaves the port in Hi-Z (the P-MOS is never activated)
  - Push-pull mode: A “0” in the Output register activates the N-MOS whereas a “1” in the Output register activates the P-MOS
- The Schmitt trigger input is activated
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state
- A read access to the output data register gets the last written value

Figure 120 shows the output configuration of the I/O port bit.

Figure 120. Output configuration



13.3.11 I/O compensation cell

This cell is used to control the I/O commutation slew rate (t_{fall} / t_{rise}) to reduce the I/O noise on power supply.

The cell is split into two blocks:

- The first block provides an optimal code for the current PVT. The code stored in this block can be read when the READY flag of the SYSCFG_CMPCR is set.
- The second block controls the I/O slew rate.The user selects the code to be applied automatically or to program it by software.

The I/O compensation cell features 2 voltage ranges: 1.62 to 2.0 V and 2.7 to 3.6 V.

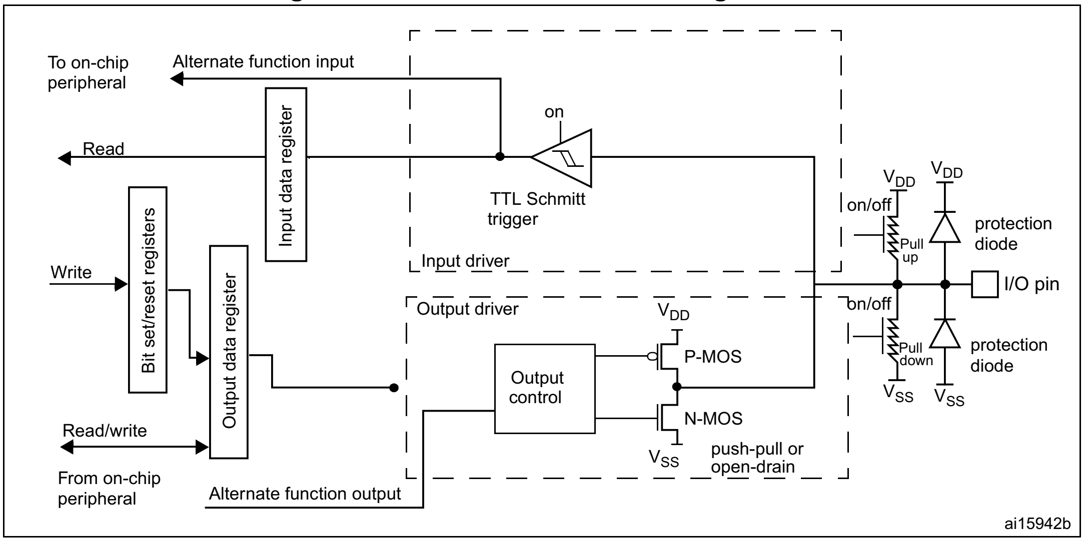
13.3.12 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- The Schmitt trigger input is activated
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle
- A read access to the input data register gets the I/O state

Figure 121 shows the Alternate function configuration of the I/O port bit.

Figure 121. Alternate function configuration



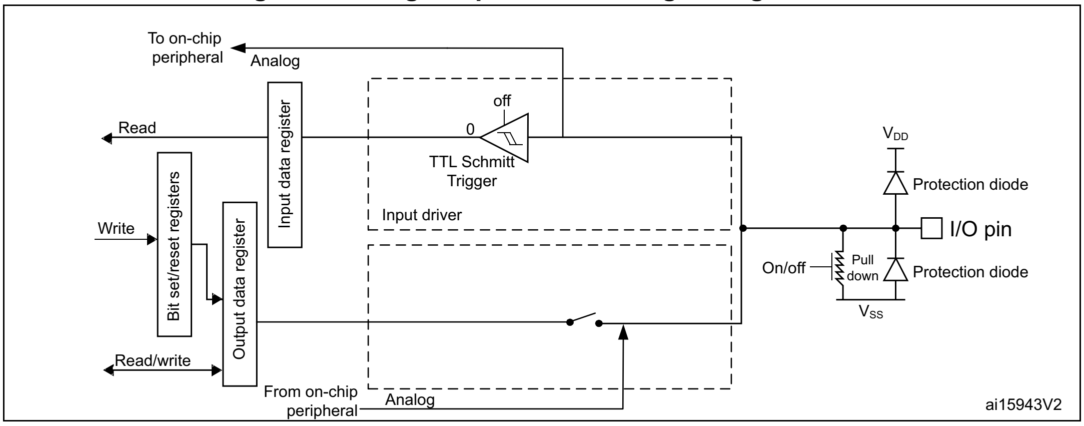
13.3.13 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up resistor is disabled by hardware.
- The weak pull-down resistor is activated or not depending on GPIOx_PUPDR value.
- Read access to the input data register gets the value “0”.

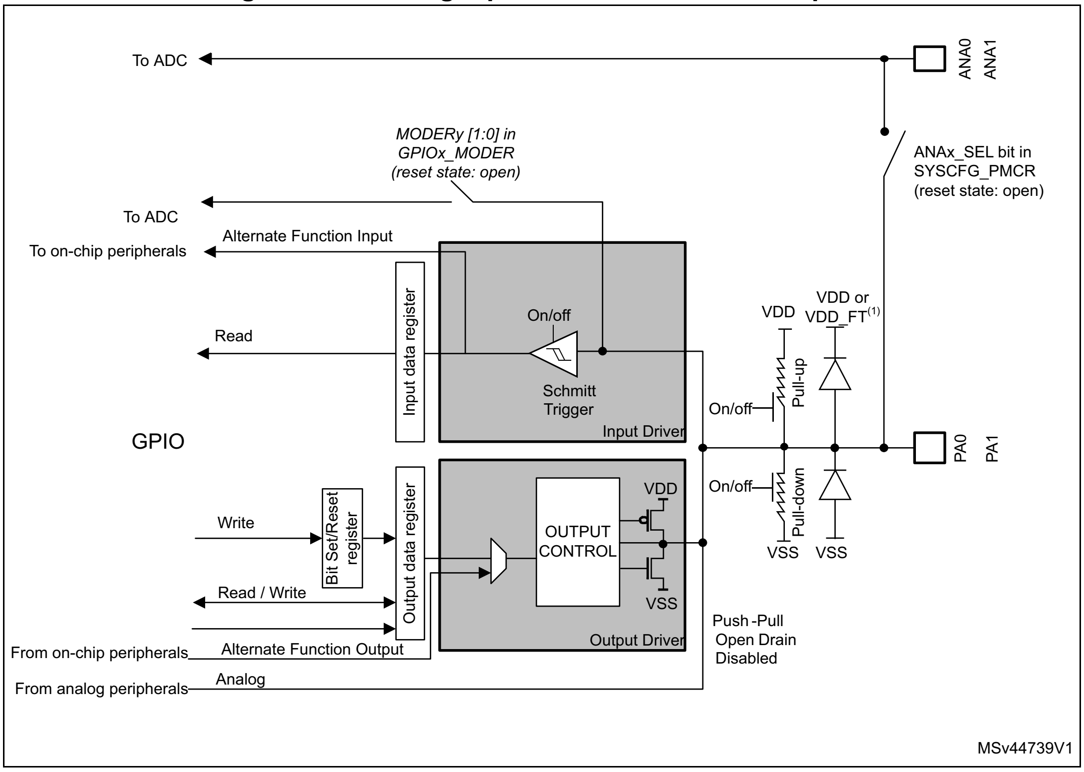
Figure 122 shows the high-impedance, analog-input configuration of the I/O port bits.

Figure 122. High impedance-analog configuration



Some pins/balls are directly connected to ANA0 and ANA1 ADC analog inputs (see *Figure 123*): there is a direct path between ANA0/ANA1 and PA0/PA1 pins/balls, through an analog switch (refer to *Section 14.3.2: SYSCFG peripheral mode configuration set register (SYSCFG_PMCSETR)* for details on how to configure analog switches).

Figure 123. Analog inputs connected to ADC inputs



1. VDD_FT is a potential specific to 5V tolerant I/Os. It is distinct from VDD.

13.3.14 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched OFF (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched ON (by setting the HSEON or LSEON bit in the RCC_CSR register) the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect. This is also true in External clock mode.

13.3.15 Using the GPIO pins in the backup supply domain

The PC13, PC14, PC15, and PI8 GPIO functionality is lost when the core supply domain is powered off (for example, when the device enters standby mode).

In this case, if their GPIO configuration (including GPIO pull-up/pull-down and open-drain settings) is not bypassed by any RTC, TAMP, or PWR (WKUP) configuration, these pins are forced into analog mode.

For details about I/O control by the RTC, refer to *Section 48.3: RTC functional description*.

13.3.16 TrustZone security (only for GPIOZ)

The TrustZone security is always activated On GPIOZ. Each I/O pin of GPIOZ port can be individually configured as secure through the GPIOZ_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function AFI, AFO, mode selection, I/O data are secure against a non-secure access:

- Input data are not redirected to another peripheral
- Output data are not replaced by another peripheral
- Secure I/O data can not be redirected to a non-secure I/O whatever the I/O is configured as alternate function or though peripherals as analog, USB, RTC, wakeup pins in PWR controller
- Non-secure I/O data can not be redirected to a secure peripheral

After reset, all GPIOZ I/O pins are secure.

Table 83 gives a summary of the I/O port secured bits following the security configuration bit in the GPIOZ_SECCFGR register.

- Secured bits: read and write operations are only allowed by a secure access. Non-secure read or write accesses on secured bits are RAZ/WI. There is no illegal access event generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.

Table 83. GPIO secured bits(1)

Secure configuration bit	Secured bit	Register name	Non-secure access on secure bits
SECy = 1 in GPIOx_SECCFGR	MODEy[1:0]	GPIOx_MODER	RAZ/WI
	OTy	GPIOx_OTYPER	
	OSPEEDy[1:0]	GPIOx_OSPEEDR	
	PUPDy[1:0]	GPIOx_PUPDR	
	IDy	GPIOx_IDR	
	ODy	GPIOx_ODR	
	BSy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	AFSELy[3:0]	GPIOx_AFRL	
	BRy	GPIOx_AFRH	
		GPIOx_BRR	

1. GPIOx, x= Z, and y=0..7.

13.4 GPIO registers

This section gives a detailed description of the GPIO registers.

For a summary of register bits, register address offsets and reset values, refer to *Table 84*.

The peripheral registers can be written in word, half word or byte mode.

13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to K, Z)

Address offset:0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODER[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode

13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to K, Z)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OT[15:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O output type.
0: Output push-pull (reset state)
1: Output open-drain

13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to K, Z)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDR[15:0][1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed
01: Medium speed
10: High speed
11: Very high speed

Note: Refer to the product datasheets for the values of OSPEEDRy bits versus V_{DD} range and external load.

13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to K, Z)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDR[15:0][1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to K, Z)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDR[15:0]:** Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to K, Z)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODR[15:0]:** Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and/or reset by writing to the GPIOx_BSRR register (x = A..F).

13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to K, Z)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BR[15:0]:** Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit
Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BS[15:0]:** Port x set I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to K, Z)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

LOCK key write sequence:

WR LCKR[16] = ‘1’ + LCKR[15:0]

WR LCKR[16] = ‘0’ + LCKR[15:0]

WR LCKR[16] = ‘1’ + LCKR[15:0]

RD LCKR

RD LCKR[16] = ‘1’ (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the lock.

After the first lock sequence on any bit of the port, any read access on the LCKK bit returns ‘1’ until the next MCU reset or peripheral reset.

Bits 15:0 **LCK[15:0]:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is ‘0.

0: Port configuration not locked

1: Port configuration locked

13.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to K, Z)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFR[7:0][3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure alternate function I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

13.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to K, Z)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFR[15:8][3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to K, Z)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BR[15:0]:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODx bit
1: Reset the corresponding ODx bit

13.4.12 GPIO secure configuration register (GPIOZ_SECCFGR)

This register provides access security control and can be written only by a secure access. It is used to configure a selected I/O as secure. A non-secure write access to this register is discarded. Read returns the current GPIOZ pin security setting.

Address offset: 0x30

Reset value: 0x0000 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	wr	rw	rw	rw

Bits 31:8 Reserved

Bits 7:0 **SEC[7:0]**: I/O pin of Port Z secure bit enable y (y= 0..7)
These bits control the security I/O port pin.
0: The I/O pin is non-secure
1: The I/O pin is secure. Refer to *Table 83: GPIO secured bits* for all corresponding secured bits.

13.4.13 GPIO hardware configuration register 10 (GPIOx_HWCFGR10) (x = A to K, Z)

Address offset: 0x3C8

Reset value: 0x0000 1240

Reset value: Block Z: 0x0001 1240

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_CFG[3:0]				SEC_CFG[3:0]			
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LOCK_CFG[3:0]				SPEED_CFG[3:0]				AF_SIZE[3:0]				AHB_IOP[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **OR_CFG[3:0]:** Option register configuration
0: Non-active
1: Active

Bits 19:16 **SEC_CFG[3:0]:** Security mechanism activation
0: Non-active
1: Active

Bits 15:12 **LOCK_CFG[3:0]:** Lock mechanism activation
0: Non-active
1: Active

Bits 11:8 **SPEED_CFG[3:0]:** Number of speed lines for each I/O
1: One line
2: Two lines

Bits 7:4 **AF_SIZE[3:0]:** Number of AF available for each I/O
Accepted value: 1 to 4
1: One AF/IO
2: Four AF/IO
3: Eight AF/IO
4: 16 AF/IO

Bits 3:0 **AHB_IOP[3:0]:** Bus interface configuration
0: AHB
1: IOP

13.4.14 GPIO hardware configuration register 9 (GPIOx_HWCFGR9) (x = A to K, Z)

Address offset: 0x3CC

Reset value: 0x0000 FFFF

Reset value: Block K: 0x0000 00FF

Reset value: Block Z: 0x0000 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EN_IO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EN_IO[15:0]:** Presence granularity, each bit indicate the presence of the IO
0: IO not present
1: IO present

13.4.15 GPIO hardware configuration register 8 (GPIOx_HWCFGR8) (x = A to K, Z)

Address offset: 0x3D0

Reset value: 0xFFFF FFFF

Reset value: Block K: 0x0000 0000

Reset value: Block Z: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AF_PRIO15[3:0]				AF_PRIO14[3:0]				AF_PRIO13[3:0]				AF_PRIO12[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AF_PRIO11[3:0]				AF_PRIO10[3:0]				AF_PRIO9[3:0]				AF_PRIO8[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **AF_PRIO15[3:0]:** Indicate the priority AF for I/O15 (0 to F)

Bits 27:24 **AF_PRIO14[3:0]:** Indicate the priority AF for I/O14 (0 to F)

Bits 23:20 **AF_PRIO13[3:0]:** Indicate the priority AF for I/O13 (0 to F)

Bits 19:16 **AF_PRIO12[3:0]:** Indicate the priority AF for I/O12 (0 to F)

Bits 15:12 **AF_PRIO11[3:0]:** Indicate the priority AF for I/O11 (0 to F)

Bits 11:8 **AF_PRIO10[3:0]:** Indicate the priority AF for I/O10 (0 to F)

Bits 7:4 **AF_PRIO9[3:0]:** Indicate the priority AF for I/O9 (0 to F)

Bits 3:0 **AF_PRIO8[3:0]:** Indicate the priority AF for I/O8 (0 to F)

13.4.16 GPIO hardware configuration register 7 (GPIOx_HWCFGR7) (x = A to K, Z)

Address offset: 0x3D4

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AF_PRIO7[3:0]				AF_PRIO6[3:0]				AF_PRIO5[3:0]				AF_PRIO4[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AF_PRIO3[3:0]				AF_PRIO2[3:0]				AF_PRIO1[3:0]				AF_PRIO0[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **AF_PRIO7[3:0]:** Indicate the priority AF for I/O7 (0 to F)

Bits 27:24 **AF_PRIO6[3:0]:** Indicate the priority AF for I/O6 (0 to F)

Bits 23:20 **AF_PRIO5[3:0]:** Indicate the priority AF for I/O5 (0 to F)

Bits 19:16 **AF_PRIO4[3:0]:** Indicate the priority AF for I/O4 (0 to F)

Bits 15:12 **AF_PRIO3[3:0]:** Indicate the priority AF for I/O3 (0 to F)

Bits 11:8 **AF_PRIO2[3:0]:** Indicate the priority AF for I/O2 (0 to F)

Bits 7:4 **AF_PRIO1[3:0]:** Indicate the priority AF for I/O1 (0 to F)

Bits 3:0 **AF_PRIO0[3:0]:** Indicate the priority AF for I/O0 (0 to F)

13.4.17 GPIO hardware configuration register 6 (GPIOx_HWCFGR6) (x = A to K, Z)

Address offset: 0x3D8

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MODER_RES[31:0]:** MODER register reset value

13.4.18 GPIO hardware configuration register 5 (GPIOx_HWCFGR5) (x = A to K, Z)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PUPDR_RES[31:0]:** Pull-up / pull-down register reset value

13.4.19 GPIO hardware configuration register 4 (GPIOx_HWCFGR4) (x = A to K, Z)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **OSPEED_RES[31:0]:** OSPEED register reset value

13.4.20 GPIO hardware configuration register 3 (GPIOx_HWCFGR3) (x = A to K, Z)

Address offset: 0x3E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OTYPER_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OTYPER_RES[31:0]:** Output type register reset value

Bits 15:0 **ODR_RES[31:0]:** Output data register reset value

13.4.21 GPIO hardware configuration register 2 (GPIOx_HWCFGR2) (x = A to K, Z)

Address offset: 0x3E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AFRL_RES[31:0]:** AF register low reset value

13.4.22 GPIO hardware configuration register 1 (GPIOx_HWCFGR1) (x = A to K, Z)

Address offset: 0x3EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH_RES[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AFRH_RES[31:0]:** AF register high reset value

13.4.23 GPIO hardware configuration register 0 (GPIOx_HWCFGR0) (x = A to K, Z)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OR_RES[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OR_RES[15:0]:** Option register reset value

13.4.24 GPIO version register (GPIOx_VERR) (x = A to K, Z)

Address offset: 0x3F4

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision
These bits return the GPIO major revision.
Major revision is 4.

Bits 3:0 **MINREV[3:0]:** Minor revision
These bits return the GPIO minor revision.
Minor revision is 0.

13.4.25 GPIO identification register (GPIOx_IPIDR) (x = A to K, Z)

Address offset: 0x3F8

Reset value: 0x000F 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPIDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPIDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPIDR[31:0]**: GPIO identifier
These bits return the GPIO identifier value.

13.4.26 GPIO size identification register (GPIOx_SIDR) (x = A to K, Z)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SIDR[31:0]**: Size identifier register
These bits return the size of the memory region allocated to GPIO registers.

13.4.27 GPIO register map

The following table gives the GPIO register map and reset values.

Table 84. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOx_MODER** (where x = **A to K, Z**)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER[1:0]		MODER0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER** (where x = **A to K, Z**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_OSPEEDR** (where x = **A to K, Z**)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR** (where x = **A to K, Z**)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = **A to K, Z**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = **A to K, Z**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = **A to K, Z**)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = **A to K, Z**)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = **A to K, Z**)	AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[3:0]				AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = **A to K, Z**)	AFR15[3:0]				AFR14[3:0]				AFR13[3:0]				AFR12[3:0]				AFR11[3:0]				AFR10[3:0]				AFR9[3:0]				AFR8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**GPIOx_BRR** (where x = **A to K, Z**))	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**GPIOZ_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																									1	1	1	1	1	1	1	1

Table 84. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03C8	**GPIOx_HWCFGR10 (where x = A to K, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_CFG[3:0]				SEC_CFG[3:0]				LOCK_CFG[3:0]				SPEED_CFG [3:0]				AF_SIZE[3:0]				AHB_IOP[3:0]			
	Reset value (for GPIOA,B,C,D, E,F,G,H,I,J,K)									0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
	Reset value (for GPIOZ)									0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0
0x03CC	**GPIOx_HWCFGR9 (where x = A to K, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_IO[15:0]															
	Reset value (for GPIOA,B,C,D, E,F,G,H,I,J)																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (for GPIOK,Z)																	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x03D0	**GPIOx_HWCFGR8 (where x = A to K, Z)**	AF_PRIO15[3:0]				AF_PRIO14[3:0]				AF_PRIO13[3:0]				AF_PRIO12[3:0]				AF_PRIO11[3:0]				AF_PRIO10[3:0]				AF_PRIO9[3:0]				AF_PRIO8[3:0]			
	Reset value (for GPIOA,B,C,D, E,F,G,H,I,J)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value (for GPIOK,Z)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03D4	**GPIOx_HWCFGR7 (where x = A to K, Z)**	AF_PRIO7[3:0]				AF_PRIO6[3:0]				AF_PRIO5[3:0]				AF_PRIO4[3:0]				AF_PRIO3[3:0]				AF_PRIO2[3:0]				AF_PRIO1[3:0]				AF_PRIO0[3:0]			
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x03D8	**GPIOx_HWCFGR6 (where x = A to K, Z)**	MODER_RES[31:16]																MODER_RES[15:0]															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x03DC	**GPIOx_HWCFGR5 (where x = A to K, Z)**	PUPDR_RES[31:16]																PUPDR_RES[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03E0	**GPIOx_HWCFGR4 (where x = A to K, Z)**	OSPEED_RES[31:16]																OSPEED_RES[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03E4	**GPIOx_HWCFGR3 (where x = A to K, Z)**	OTYPER_RES[31:16]																ODR_RES[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03E8	**GPIOx_HWCFGR2 (where x = A to K, Z)**	AFRL_RES[31:16]																AFRL_RES[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03EC	**GPIOx_HWCFGR1 (where x = A to K, Z)**	AFRH_RES[31:16]																AFRH_RES[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03F0	**GPIOx_HWCFGR0 (where x = A to K, Z)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OR_RES[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03F4	**GPIO_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	1	0	0	0	0	0	0
0x03F8	**GPIO_IPIDR**	ID[31:16]																ID[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x03FC	**GPIO_SIDR**	SID[31:16]																SID[15:0]															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

14 System configuration controller (SYSCFG)

The system configuration controller is mainly used to manage the compensation cell and other IOs and system related settings. This section applies to the whole STM32MP153 family, unless otherwise specified.

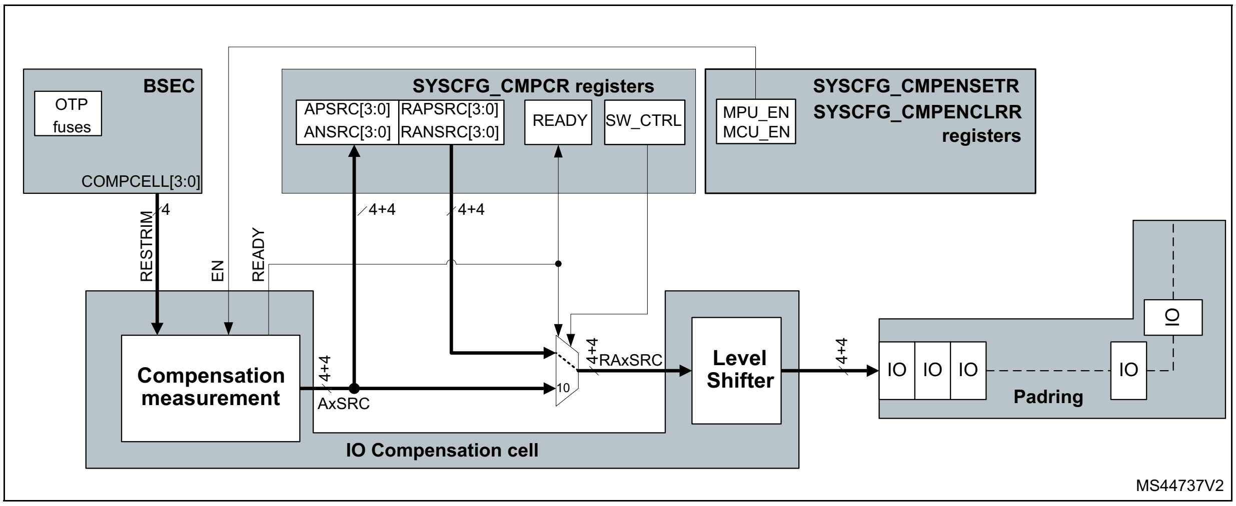
14.1 I/O compensation cell

By default the I/O compensation cell is not used. However when the I/O output buffer speed is configured in 50 MHz mode and above, it is recommended to use the compensation cell for a slew rate control on I/O $t_{f(IO)out}/t_{r(IO)out}$ commutation to reduce the I/O noise on the power supply.

Note: When enabled, the compensation cell has only impact on I/Os with GPIOx_OSPEEDRy = 0b10 or 0b11.

When the compensation cell is enabled, a READY flag is set to indicate that the compensation cell is ready and can be used, the compensation value is then continuously updated.

Figure 124. I/O compensation control overview



Sequence to enable IO compensation:

1. Ensure the CSI oscillator is enabled and ready (in RCC)
2. Set SYSCFG_CMPENSETR.MCU_EN or MPU_EN=1
3. Wait SYSCFG_CMPCR.READY = 1
4. Set SYSCFG_CMPCR.SW_CTRL = 0

Sequence to disable the IO compensation, holding the current compensation value (e.g. before going to STOP):

1. Copy actual value of SYSCFG_CMPCR.APSRC[3:0]/ANSRC[3:0] in SYSCFG_CMPCR.RAPSRC[3:0]/RANSRC[3:0]
2. Set SYSCFG_CMPCR.SW_CTRL = 1
3. Set SYSCFG_CMPENCLRR.MCU_EN and MPU_EN=1

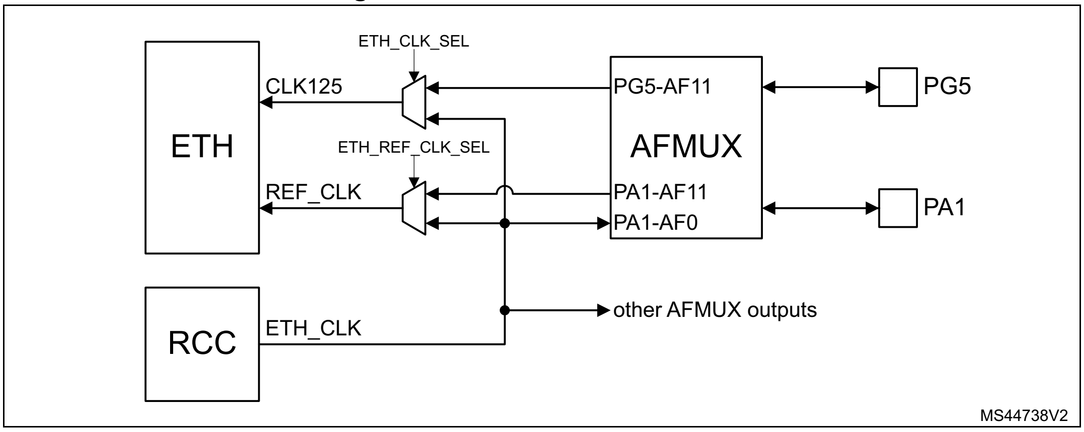
Note: the CSI oscillator is disabled automatically in Stop mode if not requested for other usages, and the CSI is always OFF in Standby mode.

14.2 Ethernet clock source

Depending on the external ethernet PHY, the ETH_CLK clock can have various usage:

- The GMII and RGMII clocks: PG5 can be used to receive either 125 MHz from PHY or internally the ETH_CLK clock at 125 MHz (ETH_CLK can be fed externally to another pin if needed).
- The RMII clock: the ETH_CLK clock at 50 MHz can be output on PA1 and at the same time used by the PHY for REF_CLK.
- ALL: the ETH_CLK clock can be used for the PHY reference clock.

Figure 125. Ethernet clock source



The ethernet clock source is controlled through the SYSCFG_PMCSETR/CLRR registers.

14.3 SYSCFG registers

14.3.1 SYSCFG boot pins control register (SYSCFG_BOOTR)

Address offset: 0x000

Reset value: 0x0000 000X

This register is used to know the state of BOOT pins and to control pull-up to reduce the static power consumption on the pin set to high level.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT2_PD	BOOT1_PD	BOOT0_PD	Res.	BOOT2	BOOT1	BOOT0
									rw	rw	rw		r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BOOT2_PD:** BOOT2 pin pull-down disable. This is used to save power in case the BOOT2 pin is connected to VDD.
- 0: (enabled) pull-down enabled. The BOOT2 pin can be left open and will take a value of 0 if open.
- 1: (disabled) pull-down disabled. The BOOT2 pin must not be left open.

Bit 5 **BOOT1_PD:** BOOT1 pin pull-down disable. This is used to save power in case the BOOT1 pin is connected to VDD
- 0: (enabled) pull-down enabled. The BOOT1 pin can be left open and will take a value of 0 if open.
- 1: (disabled) pull-down disabled. The BOOT1 pin must not be left open.

Bit 4 **BOOT0_PD:** BOOT0 pin pull-down disable. This is used to save power in case the BOOT0 pin is connected to VDD
- 0: (enabled) pull-down enabled. The BOOT0 pin can be left open and will take a value of 0 if open.
- 1: (disabled) pull-down disabled. The BOOT0 pin must not be left open.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **BOOT2:** BOOT2 pin value
- 0: (low) BOOT2 pin connected to VSS (or left open if BOOT2_PD=0)
- 1: (high) BOOT2 pin connected to VDD

Bit 1 **BOOT1:** BOOT1 pin value
- 0: (low) BOOT1 pin connected to VSS (or left open if BOOT1_PD=0)
- 1: (high) BOOT1 pin connected to VDD

Bit 0 **BOOT0:** BOOT0 pin value
- 0: (low) BOOT0 pin connected to VSS (or left open if BOOT0_PD=0)
- 1: (high) BOOT0 pin connected to VDD

14.3.2 SYSCFG peripheral mode configuration set register (SYSCFG_PMCSETR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ANA1_SEL	ANA0_SEL	ETH_SEL[2:0]			ETH_SELMII	Res.	Res.	ETH_REF_CLK_SEL	ETH_CLK_SEL
						rs	rs	rs	rs	rs	rs			rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	EN_BOOSTER	Res.	Res.	I2C6_FMP	I2C5_FMP	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP
						rs	rs			rs	rs	rs	rs	rs	rs

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **ANA1_SEL**: controls analog connection between the ANA1 and PA1 pin.
Set by software.
0: Writing '0' has no effect, reading '0' means no connection. For best ADC performance the ANA1 input must be used for package having ANA1 pin present.
1: Writing '1' set this bit, reading '1' means ANA1 internally connected to PA1. Must only be used on package not having ANA1 pin.

Bit 24 **ANA0_SEL**: controls analog connection between ANA0 and PA0 pin.
Set by software.
0: Writing '0' has no effect, reading '0' means no connection. For best ADC performance ANA0 input must be used for package having ANA0 pin present.
1: Writing '1' set this bit, reading '1' means ANA0 internally connected to PA0. Must only be used on package not having ANA0 pin.

Bits 23:21 **ETH_SEL[2:0]**: Ethernet PHY interface selection.
Set by software. Writing bit as '0' has no effect, set individual bits by writing '1', reading means
000: GMII or MII
001: RGMII
010: Reserved
011: Reserved
100: RMII
101: Reserved
110: Reserved
111: Reserved
Note: Configuration must be done while the ETH is under reset and before enabling the ETH clocks

Bit 20 **ETH_SELMII**: controls MII or GMII when ETH_SEL[2:0] = 0b000.
Set by software.
0: (GMII) Writing '0' has no effect, reading '0' means MII/GMII clock mux controlled by ETH GMAC (GMII PHY use case)
1: (MII) Writing '1' set this bit, reading '1' means MII/GMII clock mux forced in MII mode (MII PHY use case)

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **ETH_REF_CLK_SEL**: Ethernet 50MHz RMII clock selection.

Set by software.

0: Writing '0' has no effect, reading '0' means External clock is used. Need selection of AFMux. Could be used with all PHY

1: Writing '1' set this bit, reading '1' means Internal clock ETH_CLK1 from RCC is used regardless AFMux. Could be used only with RMII PHY.

Bit 16 **ETH_CLK_SEL**: Gigabit Ethernet 125 MHz clock selection. Only useful for GMII or RGMII PHY

Set by software.

0: Writing '0' has no effect, reading '0' means External clock is used. Need selection of AFMux

1: Writing '1' set this bit, reading '1' means Internal clock ETH_CLK1 from RCC is used regardless AFMux

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ANASWVDD**: GPIO analog switches control voltage selection

Set by software.

0: Writing '0' has no effect, reading '0' means IOs analog switches supplied by VDDA (**EN_BOOSTER**=0) or Booster output (**EN_BOOSTER**=1)

1: Writing '1' set this bit, reading '1' means IOs analog switches supplied by VDD (regardless **EN_BOOSTER** bit, which should be cleared to avoid unwanted power consumption). Useful to avoid using Booster (add current consumption) when VDDA < 2.7V, but VDD > 2.7V. When both VDD < 2.7V and VDDA < 2.7V, the Booster is still needed to get full AC performances from IOs analog switches.

Bit 8 **EN_BOOSTER**: Pad Booster enable.

Set by software.

used to reduce the Total Harmonic Distortion of the IOs analog switches when VDDA and VDD supplies are below 2.7V. Activating the booster allows to guaranty the AC performance on IOs analog switches. When activated the performance of the analog switch is the same on the full voltage range. This bit is only useful when **ANASWVDD**=0

0: Writing '0' has no effect, reading '0' means Booster is disabled

1: Writing '1' set this bit, reading '1' means Booster is enabled. Note that this will add a current consumption on VDD

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **I2C6_FMP**: Fast Mode Plus (FM+) enable

Set by software.

0: Writing '0' has no effect, reading '0' means I2C6 usage is possible up to 400 kHz

1: Writing '1' set this bit, reading '1' means the I2C6 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 4 **I2C5_FMP**: Fast Mode Plus (FM+) Enable

Set by software.

0: Writing '0' has no effect, reading '0' means I2C5 usage is possible up to 400 kHz

1: Writing '1' set this bit, reading '1' means the I2C5 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 3 **I2C4_FMP**: Fast Mode Plus (FM+) Enable

Set by software.

0: Writing '0' has no effect, reading '0' means I2C4 usage is possible up to 400 kHz

1: Writing '1' set this bit, reading '1' means the I2C4 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1MHz.

Bit 2 **I2C3_FMP**: Fast Mode Plus (FM+) Enable

Set by software.

0: Writing '0' has no effect, reading '0' means I2C3 usage is possible up to 400kHz

1: Writing '1' set this bit, reading '1' means the I2C3 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 1 **I2C2_FMP**: Fast Mode Plus (FM+) Enable

Set by software.

0: Writing '0' has no effect, reading '0' means I2C2 usage is possible up to 400 kHz

1: Writing '1' set this bit, reading '1' means the I2C2 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 0 **I2C1_FMP**: Fast Mode Plus (FM+) Enable

Set by software.

0: Writing '0' has no effect, reading '0' means I2C1 usage is possible up to 400 kHz

1: Writing '1' set this bit, reading '1' means the I2C1 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

14.3.3 SYSCFG peripheral mode configuration clear register (SYSCFG_PMCCLRR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ANA1_SEL	ANA0_SEL	ETH_SEL[2:0]			ETH_SELMII	Res.	Res.	ETH_REF_CLK_SEL	ETH_CLK_SEL
						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	EN_BOOSTER	Res.	Res.	I2C6_FMP	I2C5_FMP	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP
						rc_w1	rc_w1			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **ANA1_SEL**: controls analog connection between the ANA1 and PA1 pin.
Cleared by software.
0: Writing '0' has no effect, reading '0' means no connection. For best ADC performance the ANA1 input must be used for package having ANA1 pin present.
1: Writing '1' clear this bit, reading '1' means ANA1 internally connected to PA1. Must only be used on package not having ANA1 pin.

Bit 24 **ANA0_SEL**: controls analog connection between ANA0 and PA0 pin.
Cleared by software.
0: Writing '0' has no effect, reading '0' means no connection. For best ADC performance ANA0 input must be used for package having ANA0 pin present.
1: Writing '1' clear this bit, reading '1' means ANA0 internally connected to PA0. Must only be used on package not having ANA0 pin.

Bits 23:21 **ETH_SEL[2:0]**: Ethernet PHY interface selection.
Cleared by software. Writing bit as '0' has no effect, clear individual bits by writing ‘1’, reading means
000: GMII or MII
001: RGMII
010: Reserved
011: Reserved
100: RMII
101: Reserved
110: Reserved
111: Reserved
Note: Configuration must be done while the ETH is under reset and before enabling the ETH clocks

Bit 20 **ETH_SELMII**: controls MII or GMII when ETH_SEL[2:0] = 0b000.
Cleared by software.
0: (GMII) Writing '0' has no effect, reading '0' means MII/GMII clock mux controlled by ETH GMAC (GMII PHY use case)
1: (MII) Writing '1' clear this bit, reading '1' means MII/GMII clock mux forced in MII mode (MII PHY use case)

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **ETH_REF_CLK_SEL**: Ethernet 50MHz RMII clock selection.
Cleared by software.
0: Writing '0' has no effect, reading '0' means External clock is used. Need selection of AFMux. Could be used with all PHY
1: Writing '1' clear this bit, reading '1' means Internal clock ETH_CLK1 from RCC is used regardless AFMux. Could be used only with RMII PHY.

Bit 16 **ETH_CLK_SEL**: Gigabit Ethernet 125 MHz clock selection. Only useful for GMII or RGMII PHY
Cleared by software.
0: Writing '0' has no effect, reading '0' means External clock is used. Need selection of AFMux
1: Writing '1' clear this bit, reading '1' means Internal clock ETH_CLK1 from RCC is used regardless AFMux

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ANASWVDD**: GPIO analog switches control voltage selection

Cleared by software.

0: Writing '0' has no effect, reading '0' means IOs analog switches supplied by VDDA (**EN_BOOSTER**=0) or Booster output (**EN_BOOSTER**=1)

1: Writing '1' clear this bit, reading '1' means IOs analog switches supplied by VDD (regardless **EN_BOOSTER** bit, which should be cleared to avoid unwanted power consumption). Useful to avoid using Booster (add current consumption) when VDDA < 2.7V, but VDD > 2.7V.

Bit 8 **EN_BOOSTER**: Pad Booster enable.

Cleared by software.

used to reduce the Total Harmonic Distortion of the IOs analog switches when VDDA and VDD supplies are below 2.7V. Activating the booster allows to guaranty the AC performance on IOs analog switches. When activated the performance of the analog switch is the same on the full voltage range. This bit is only useful when **ANASWVDD**=0

0: Writing '0' has no effect, reading '0' means Booster is disabled

1: Writing '1' clear this bit, reading '1' means Booster is enabled. Note that this will add a current consumption on VDD.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **I2C6_FMP**: Fast Mode Plus (FM+) enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means I2C6 usage is possible up to 400 kHz

1: Writing '1' clear this bit, reading '1' means the I2C6 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 4 **I2C5_FMP**: Fast Mode Plus (FM+) Enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means I2C5 usage is possible up to 400 kHz

1: Writing '1' clear this bit, reading '1' means the I2C5 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 3 **I2C4_FMP**: Fast Mode Plus (FM+) Enable

Cleared by software.

0: Writing '0' has no effect, reading '0' means I2C4 usage is possible up to 400 kHz

1: Writing '1' clear this bit, reading '1' means the I2C4 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1MHz.

Bit 2 **I2C3_FMP**: Fast Mode Plus (FM+) Enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means I2C3 usage is possible up to 400kHz
1: Writing '1' clear this bit, reading '1' means the I2C3 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 1 **I2C2_FMP**: Fast Mode Plus (FM+) Enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means I2C2 usage is possible up to 400 kHz
1: Writing '1' clear this bit, reading '1' means the I2C2 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

Bit 0 **I2C1_FMP**: Fast Mode Plus (FM+) Enable
Cleared by software.
0: Writing '0' has no effect, reading '0' means I2C1 usage is possible up to 400 kHz
1: Writing '1' clear this bit, reading '1' means the I2C1 SCL & SDA pads in Fast Mode Plus (FM+), as soon as the correspondent alternate function is selected. The pad low drive is enhanced to allow usage up to 1 MHz.

14.3.4 SYSCFG IO control set register (SYSCFG_IOCTRLSETR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLVEN _SPI	HSLVEN_ SDMMC	HSLVEN _ETH	HSLVEN_ QUADSPI	HSLVEN _TRACE
											rs	rs	rs	rs	rs

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **HSLVEN_SPI**: High Speed Low Voltage Pad mode Enable.
Set by software.
Controls the speed of _h and _e pads when a SPIy_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' enables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 3 **HSLVEN_SDMMC**: High Speed Low Voltage Pad mode Enable.
Set by software.
Controls the speed of _h and _e pads when a SDMMCy_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' enables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 2 **HSLVEN_ETH**: High Speed Low Voltage Pad mode Enable.
Set by software.
Controls the speed of _h and _e pads when a ETH_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' enables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 1 **HSLVEN_QUADSPI**: High Speed Low Voltage Pad mode Enable.
Set by software.
Controls the speed of _h and _e pads when a QUADSPI_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' enables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 0 **HSLVEN_TRACE**: High Speed Low Voltage Pad mode Enable.
Set by software.
Controls the speed of _h and _e pads when a TRACEx signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' enables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

1. This bit is not taken into account if the OTP bit **(product_below_2V5)** is 0 (default)
2. Enabling High Speed mode while VDD > 2.7V can damage the IC. See pad characteristics for details.

Danger: **Enabling High Speed mode while VDD > 2.7V can damage the IC.**
See pad characteristics for details

14.3.5 SYSCFG IO control clear register (SYSCFG_IOCTRLCLRR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLVEN_SPI	HSLVEN_SDMMC	HSLVEN_ETH	HSLVEN_QUADSPI	HSLVEN_TRACE
											rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **HSLVEN_SPI**: High Speed Low Voltage Pad mode Enable.
Cleared by software.
Controls the speed of _h and _e pads when a SPIy_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' disables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 3 **HSLVEN_SDMMC**: High Speed Low Voltage Pad mode Enable.
Cleared by software.
Controls the speed of _h and _e pads when a SDMMCy_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' disables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 2 **HSLVEN_ETH**: High Speed Low Voltage Pad mode Enable.
Cleared by software.
Controls the speed of _h and _e pads when a ETH_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' disables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 1 **HSLVEN_QUADSPI**: High Speed Low Voltage Pad mode Enable.
Cleared by software.
Controls the speed of _h and _e pads when a QUADSPI_x signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' disables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

Bit 0 **HSLVEN_TRACE**: High Speed Low Voltage Pad mode Enable.
Cleared by software.
Controls the speed of _h and _e pads when a TRACEx signal is selected in AFMUX.
0: Writing '0' has no effect, reading '0' means High Speed mode disabled
1: Writing '1' disables High Speed mode, reading '1' means High Speed mode enabled(1)(2)

1. This bit is not taken into account if the OTP bit **(product_below_2V5)** is 0 (default)
2. Enabling High Speed mode while VDD > 2.7V can damage the IC. See pad characteristics for details.

Danger: **Enabling High Speed mode while VDD > 2.7V can damage the IC.**
See pad characteristics for details

14.3.6 SYSCFG interconnect control register (SYSCFG_ICNR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	AXI_M10	AXI_M9	AXI_M8	AXI_M7	AXI_M6	AXI_M5	Res.	AXI_M3	AXI_M2	AXI_M1	AXI_M0
					rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **AXI_M10:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 9 **AXI_M9:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 8 **AXI_M8:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 7 **AXI_M7:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 6 **AXI_M6:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 5 **AXI_M5:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 4 Reserved, must be kept at reset value.

Bit 3 **AXI_M3:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning

0: (S0) Master access DDR through Slave S0

1: (S1) Master access DDR through Slave S1

Bit 2 **AXI_M2:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning
0: (S0) Master access DDR through Slave S0
1: (S1) Master access DDR through Slave S1

Bit 1 **AXI_M1:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning
0: (S0) Master access DDR through Slave S0
1: (S1) Master access DDR through Slave S1

Bit 0 **AXI_M0:** controls which slave port is used by the master to access the DDR. Allows the performance/latency tuning
0: (S0) Master access DDR through Slave S0
1: (S1) Master access DDR through Slave S1

14.3.7 SYSCFG compensation cell control register (SYSCFG_CMPCR)

Address offset: 0x020

Reset value: 0xXX87 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSRC[3:0]				ANSRC[3:0]				RAPSRC[3:0]				RANSRC[3:0]			
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	Res.	Res.	Res.	Res.	Res.	Res.	SW_CTRL	Res.
							r							rw	

Bits 31:28 **APSRC[3:0]**: PMOS I/O Compensation value provided by compensation cell.
Value sent to IOs compensation when SW_CTRL = 0 and READY = 1

Bits 27:24 **ANSRC[3:0]**: NMOS I/O Compensation value provided by compensation cell.
Value sent to IOs when SW_CTRL = 0 and READY = 1

Bits 23:20 **RAPSRC[3:0]**: PMOS I/O Compensation value sent to IOs when SW_CTRL = 1
0000: reserved
0001: maximum compensation of fast conditions
....
1000: compensation for typical conditions
....
1110: maximum compensation of slow conditions
1111: reserved
Note: If compensation is needed, it is recommended to use automatic compensation

Bits 19:16 **RANSRC[3:0]**: NMOS I/O Compensation value sent to IOs when SW_CTRL = 1
0000: reserved
0001: maximum compensation of slow conditions
....
0111: compensation for typical conditions
....
1110: maximum compensation of fast conditions
1111: reserved
Note: If compensation is needed, it is recommended to use automatic compensation

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **READY**: Compensation cell ready flag
0: I/O compensation cell not ready.
1: I/O compensation cell ready, the values of APSRC[3:0] and ANSRC[3:0] are valid

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **SW_CTRL**: Compensation Software Control
0: IO compensation values come from compensation values in ANSRC[3:0] and APSRC[3:0]
1: IO compensation values come from RANSRC[3:0] and RAPSRC[3:0] register values
Note: SW_CTRL = 0 is not taken into account until READY = 1.
This means that whenever SW_CTRL, the RANSRC[3:0] and RAPSRC[3:0] are used for IO compensation when the compensation cell is in power down (MCU_EN = 0 and MPU_EN = 0, which is the case after a reset).

Bit 0 Reserved, must be kept at reset value.

14.3.8 SYSCFG compensation cell enable set register (SYSCFG_CMPENSETR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCU_EN	MPU_EN
														rs	rs

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCU_EN**: Compensation cell enable

Set by software.

0: (power_down) Writing '0' has no effect, reading ‘0’ mean I/O compensation cell in power-down mode if MPU_EN=0

1: (enabled) Writing '1' enable I/O compensation cell, reading ‘1’ mean /O compensation cell is enabled

Note: The CSI oscillator must be enabled and ready (controlled in RCC) before MPU_EN could be set to 1. Similarly, the CSI oscillator could be disabled only if MPU_EN and MCU_EN are set to 0.

Bit 0 **MPU_EN**: Compensation cell enable

Set by software.

0: (power_down) Writing '0' has no effect, reading ‘0’ mean I/O compensation cell in power-down mode if MCU_EN=0

1: (enabled) Writing '1' enable I/O compensation cell, reading ‘1’ mean /O compensation cell is enabled

Note: The CSI oscillator must be enabled and ready (controlled in RCC) before MPU_EN could be set to 1. Similarly, the CSI oscillator could be disabled only if MPU_EN and MCU_EN are set to 0.

14.3.9 SYSCFG compensation cell enable clear register (SYSCFG_CMPENCLRR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCU_EN	MPU_EN
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **MCU_EN**: Compensation cell enable

Cleared by software.

0: (power_down) Writing '0' has no effect, reading ‘0’ mean I/O compensation cell in power-down mode if MPU_EN=0

1: (enabled) Writing '1' clear MCU_EN bit, reading ‘1’ mean /O compensation cell is enabled

Note: The CSI oscillator must be enabled and ready (controlled in RCC) before MPU_EN could be set to 1. Similarly, the CSI oscillator could be disabled only if MPU_EN and MCU_EN are set to 0.

Bit 0 **MPU_EN**: Compensation cell enable

Cleared by software.

0: (power_down) Writing '0' has no effect, reading ‘0’ mean I/O compensation cell in powerdown mode if MCU_EN=0

1: (enabled) Writing '1' clear MPU_EN bit, reading ‘1’ mean /O compensation cell is enabled

Note: The CSI oscillator must be enabled and ready (controlled in RCC) before MPU_EN could be set to 1. Similarly, the CSI oscillator could be disabled only if MPU_EN and MCU_EN are set to 0.

14.3.10 SYSCFG control timer break register (SYSCFG_CBR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDL	Res.	CLL
													rs		rs

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **PVDL**: PVD lock enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/8/15/16/17 Break input, as well as the PVDE and PLS[2:0] in the PWR_CR1 register.

0: PVD interrupt disconnected from TIM1/8/15/16/17 Break input. PVDE and PLS[2:0] bits can be programmed by the application.

1: PVD interrupt connected to TIM1/8/15/16/17 Break input. PVDE and PLS[2:0] bits are read only.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CLL**: Cortex-M4 LOCKUP (Hardfault) output enable bit

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the connection of Cortex-M4 LOCKUP (Hardfault) output to TIM1/8/15/16/17 Break input

0: Cortex-M4 LOCKUP output disconnected from TIM1/8/15/16/17 Break inputs

1: Cortex-M4 LOCKUP output connected to TIM1/8/15/16/17 Break inputs

14.3.11 SYSCFG version register (SYSCFG_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision
These bits return the SYSCFG major revision.
Major revision is 2.

Bits 3:0 **MINREV[3:0]:** Minor revision
These bits return the SYSCFG minor revision.
Minor revision is 0.

14.3.12 SYSCFG identification register (SYSCFG_IPIDR)

Address offset: 0x03F8

Reset value: 0x0003 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: SYSCFG identifier
These bits return the SYSCFG identifier value.

14.3.13 SYSCFG size identification register (SYSCFG_SIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits return the size of the memory region allocated to SYSCFG registers.

14.3.14 SYSCFG interface register map

Table 85 gives the SYSCFG interface register map and reset values.

Table 85. SYSCFG interface register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SYSCFG_BOOTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT2_PD	BOOT1_PD	BOOT0_PD	Res.	BOOT2	BOOT1	BOOT0
	Reset value																										0	0	0		-	-	-
0x004	SYSCFG_PMCSETR	Res.	Res.	Res.	Res.	Res.	Res.	ANA1_SEL	ANA0_SEL	ETH_SEL[2:0]			ETH_SELMII	Res.	Res.	ETH_REF_CLK_SEL	ETH_CLK_SEL	Res.	Res.	Res.	Res.	Res.	Res.	ANASWVDD	EN_BOOSTER	Res.	Res.	I2C6_FMP	I2C5_FMP	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP
	Reset value							0	0	0	0	0	0			0	0							0	0			0	0	0	0	0	0
0x008 - 0x014	Res.erved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x018	SYSCFG_ IOCTRLSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLVEN_SPI	HSLVEN_SDMMC	HSLVEN_ETH	HSLVEN_QUADSPI	HSLVEN_TRACE
	Reset value																												0	0	0	0	0
0x01C	SYSCFG_ICNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXI_M10	AXI_M9	AXI_M8	AXI_M7	AXI_M6	AXI_M5	Res.	AXI_M3	AXI_M2	AXI_M1	AXI_M0
	Reset value																						0	0	0	0	0	0		0	0	0	0
0x020	SYSCFG_CMPCR	APSRC[3:0]				ANSRC[3:0]				RAPSRC[3:0]				RANSRC[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	Res.	Res.	Res.	Res.	Res.	Res.	SW_CTR	CMP_PD
	Reset value	-	-	-	-	-	-	-	-	1	0	0	0	0	1	1	1								0							0	0
0x024	SYSCFG_CMPENSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCU_EN	MPU_EN
	Reset value																															0	0
0x028	SYSCFG_CMPENCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCU_EN	MPU_EN
	Reset value																															0	0
0x02C	SYSCFG_CBR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDL	Res.	CLL
	Reset value																														0		0
0x030 - 0x040	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 85. SYSCFG interface register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044	SYSCFG_PMCCLRR	Res.	Res.	Res.	Res.	Res.	Res.	ANA1_SEL	ANA0_SEL	ETH_SEL[2:0]			ETH_SELMII	Res.	Res.	ETH_REF_CLK_SEL	ETH_CLK_SEL	Res.	Res.	Res.	Res.	Res.	Res.	ANASWDD	EN_BOOSTER	Res.	Res.	I2C6_FMP	I2C5_FMP	I2C4_FMP	I2C3_FMP	I2C2_FMP	I2C1_FMP
	Reset value							0	0	0	0	0	0			0	0							0	0			0	0	0	0	0	0
0x048 - 0x054	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x058	SYSCFG_IOCTRLCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLVEN_SPI	HSLVEN_SDMMC	HSLVEN_ETH	HSLVEN_QUADSPI	HSLVEN_TRACE
	Reset value																												0	0	0	0	0
0x05C - 0x03F0	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x3F4	SYSCFG_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	0	0	0	0	0
0x3F8	SYSCFG_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3FC	SYSCFG_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

15 Extended TrustZone® protection controller (ETZPC)

15.1 ETZPC introduction

The ETZPC configures TrustZone security in a SoC having bus masters and slaves with programmable-security attributes (securable resources) such as:

- On-chip RAM/ROM with programmable secure region size
- AHB and APB peripherals to be made secure
- AHB and APB peripherals to be set as isolated

15.2 ETZPC features

- 32-bit APB4 interface
- ETZPC is always Write-secure
- Register set to control SoC security settings for:
  - securable on-chip RAM and ROM secure region size
  - access rights for securable AHB and APB peripherals
  - resource isolation to Cortex-M4 domain for AHB and APB peripherals
- Locking of the security configuration per memory region and per peripheral
- Configurable memory region size in 4-Kbyte steps.

15.3 Extended TrustZone architecture and ETZPC

- The Arm TrustZone model supports the isolation between:
  - A secure world, where security sensitive applications are run and critical resources are located; secure transactions are signaled with AxPROT[1] = 0 on the AXI bus
  - A non-secure or public world (Linux user space); non-secure transactions are signaled with AxPROT[1] = 1 on the AXI bus.
  - The STM32MP153 TrustZone architecture is extended beyond AXI and Cortex-A with AXI/AHB sideband signals to propagate the secure/non-secure transaction attributes to AHB and APB agents.
- AHB and APB peripherals are categorized as:
  - **secure**:
    These peripherals are protected by an AHB/APB firewall stub used to reject any non-secure access.
    Non secure writes are ignored with a Bus error.
    Non secure reads do not proceed to the peripheral, zeros are returned on the bus with a bus error.
  - **securable**:
    These peripherals are protected by an AHB/APB firewall, which is controlled by ETZPC to define its security mode.
    When the peripheral is configured as secure by ETZPC: non-secure writes are ignored with a Bus error, non-secure reads do not proceed to the peripheral, and zeros are returned on the bus with a Bus error. Only secure reads and writes can

proceed to the peripheral.
When the peripheral is configured as non-secure by the ETZPC, both secure and non-secure reads and writes can proceed to the peripheral.

– **non-secure:**
These peripherals are connected directly to AHB/APB and both secure and non-secure read and write can proceed to the peripheral.

– **TrustZone-aware**:
These peripherals are connected directly to the AHB or APB4 bus and implement a specific TrustZone behavior, for example a subset of registers can be made secure according to the peripheral state, or to its programming interface.

Note: The Bus error is conditioned to interconnect clocking (for example, illicit access to an un-clocked peripheral does not generate a bus error).

- AXI bus masters drive the AxPROT[1] signal according to their secure state; an AXI bus master can be:
  a) non-secure (in which case AxPROT[1]=1)
  b) TrustZone capable (Cortex-A7,MDMA) in which case AxPROT[1] is driven low when the bus master is in secure state.
- AHB bus masters are normally non-secure.

  It is expected that the ETZPC is programmed during secure BOOT and not changed later:
  – The TZMA secure region size can only be changed by secure software.

Note: Before releasing a RAM region to the non-secure world by programming the TZMA to decrease the secure region size, the RAM should be cleared.

15.4 ETZPC functional description

The ETZPC consists of a set of registers to control securable IPs through:

- **r0size[]** bits to define each TZMA secure region size.
  decprot[1:0] bits to control each AHB/APB securable peripheral access permissions as:
  - decprot[1:0] = 00: the peripheral is secure; read/write accesses are allowed by the secure world only.
  - decprot[1:0] = 01: the peripheral is write secure; read accesses are allowed by both secure and non-secure worlds, but write accesses are allowed by the secure world only.
  - decprot[1:0] = 10: The peripheral is non-secure and certain MCU peripherals are isolated, according to *Table 91: decprot bits for isolable peripherals (type 2) and behavior*.
  - decprot[1:0] = 11: the peripheral is non-secure; both read and write access are allowed by both the secure and non-secure worlds.

By default, after reset, memory regions are secure and all peripherals are read/ write secure.

r0size[] and **decprot[]** can be individually locked to prevent security configuration changes until next reset.

15.5 STM32MP153 security architecture

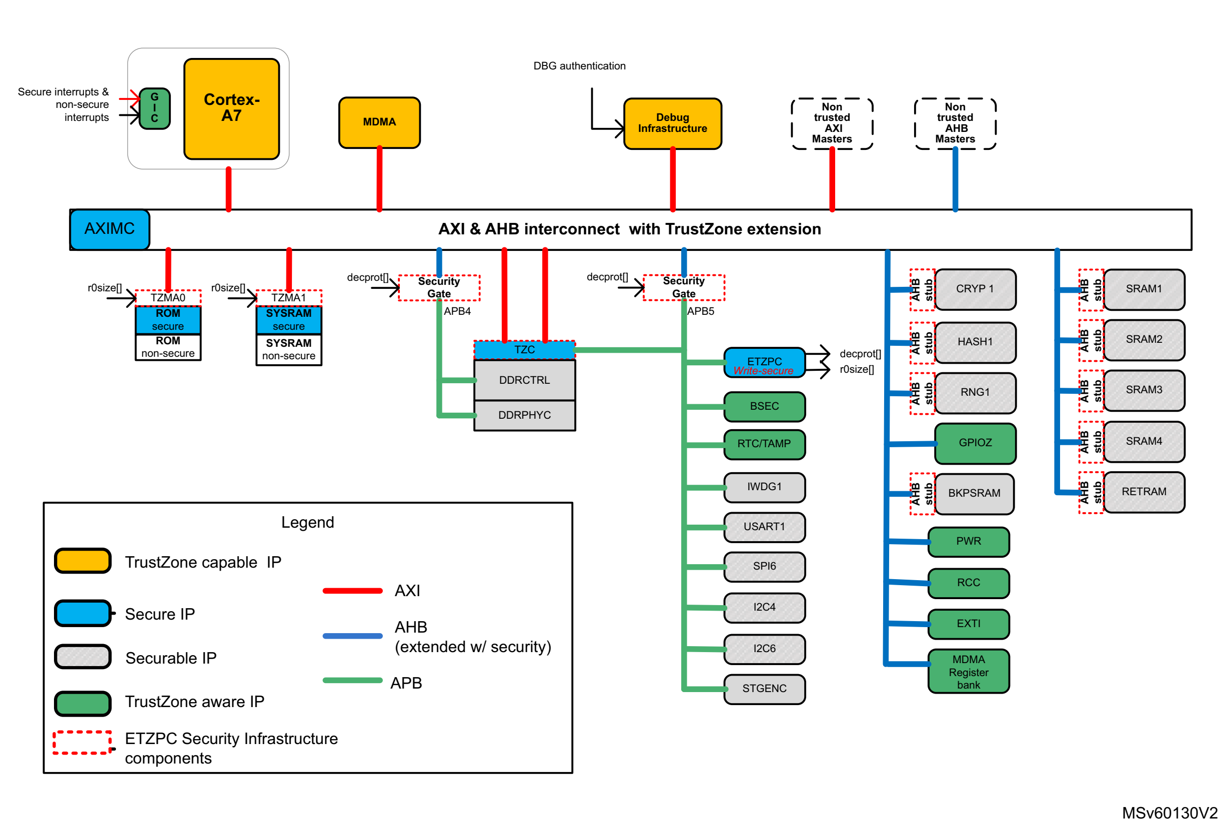
IP security properties can be classified as:

- TrustZone capable bus master IPs, as listed in *Table 89*.
- TrustZone aware peripherals, whose behavior is conditioned to transaction security, as listed in *Table 87*.
- TrustZone securable peripherals which rely on some AHB stub or on the AHB2APB bridge to control IP access according to decprot[] bits and transaction security. These are listed in *Table 86: STM32MP153 secure and securable peripherals* with type P.
- TrustZone secure (always secure) peripherals, as listed in *Table 86: STM32MP153 secure and securable peripherals* with type S or WS (write secure).

Note: Debug access port (DAP) security properties are set via the DBG authentication interface.

The STM32MP153 security architecture is shown in *Figure 126*.

Figure 126. STM32MP153 security architecture



The STM32MP153 security architecture is based on an extended version of Arm TrustZone with:

- TrustZone extension to AHB
- AHB stubs to protect securable AHB peripherals
- AHB2APB bridge using PPROT[1] signal propagation to TrustZone aware IPs listed in *Table 87*
- AHB2APB bridge to protect the securable APB peripherals listed in *Table 86: STM32MP153 secure and securable peripherals*
- TZMA firewalls for ROM and SYSRAM shown in *Table 88*

Table 86. STM32MP153 secure and securable peripherals

IP	Type(1)	Bus	comment
DDRPHYC	P	APB4	DDR PHY
DDRCTRL	P	APB4	DDR controller
CRYP1	P	AHB	CRYP1
HASH1	P	AHB	HASH1
RNG1	P	AHB	RNG1
I2C4	P	APB5	I2C for secure applications
I2C6	P	APB5	I2C for secure applications

Table 86. STM32MP153 secure and securable peripherals (continued)

IP	Type[1]	Bus	comment
SPI6	P	APB5	SPI for secure applications
USART1	P	APB5	USART for secure applications
IWDG1	P	APB5	Secure Independent Watchdog
BKPSRAM	P	AHB	BKPSRAM (4 Kbytes)
STGENC	P	APB5	System Timer GENerator (STGEN) Control port
TZC	S	APB5	TrustZone address space controller
ETZPC	WS	APB5	TrustZone protection controller (this IP)
SRAM1/2/3/4	P	AHB	SRAM1/2/3/4
RETRAM	P	AHB	RETRAM
AXIM	S	AXI	GPV local to AXIM

1. P: securable peripherals, S: R/W secure peripheral, WS: Write secure peripheral.

Table 87. STM32MP153 TrustZone-aware peripherals

IP	Bus	comment
BSEC	APB5	BSEC includes a subset of secure registers
MDMA	AHB	MDMA supports security per channel and its slave port is made TrustZone aware to forbid non-secure access to secure channels
GPIOZ	AHB	GPIOz is TrustZone-aware and supporting pin level security
RCC	AHB	RCC includes sensitive registers which are write secure
PWR	AHB	PWR includes sensitive registers which are write secure
EXTI	AHB	EXTI includes sensitive registers which are write secure
RTC and Tamper	APB	RTC/TAMP includes a subset of secure registers

For more information about IP security features refer to the corresponding IP section.

Table 88. STM32MP153 memory firewalls

On-chip memory	IP	comment
ROM	TZMA0	ROM can be partitioned into secure/non-secure regions on a 4 Kbyte boundary, the lower region is secure with 0 K, 4 K, 8 K, and so on up to 128 Kbyte size. After system reset all ROM is secure.
SYSRAM	TZMA1	SYSRAM can be partitioned into secure/non-secure regions at a 4 Kbyte boundary, the lower region is secure with 0 K, 4 K, 8 K, and so on up to 256 Kbyte size. After system reset all SYSRAM is secure.

Table 89. STM32MP153 TrustZone-capable bus masters

IP	I/F	comment
Cortex-A7	AXI	Cortex-A7 is TrustZone capable.
MDMA	AXI	MDMA channels can be set as secure or non-secure.
DAP AXI-AP	AXI	AXI-AP is made TustZone aware according to debug authentication interface (spiden, spniden) controlled by the BSEC.

15.6 STM32MP153 MCU resource isolation

The ETZPC can be used to define a set of resources to be accessed only by the Cortex-M4 domain (MCU resource isolation). Besides the Cortex-M4, the domain may also include several DMA masters assigned to the MCU isolation domain.

Resources and DMA masters that can be assigned to MCU isolation domains are listed in *Table 93: Programmable options according to resource type* with type =2 and type =3. This concerns a large set of resources, with access restricted to the Cortex-M4 and DMA master under its control for application safety purposes.

Resource isolation is programmed by the decprot[] bits and access filtering using AHB stubs and AHB2APB firewalling according to the Master IDs. Because security and MCU resource isolation peripherals are not orthogonal, the resources listed in *Table 93* are of 3 types as follows:

- **type 1**: securable peripherals
  – the decprot[] bits define access rules according to*Table 90*
  – failed access are reported with a bus error
- **type 2**: isolable only peripherals
  – the decprot[] bits define access rules according to *Table 91*
  – failed access are reported with a bus error
- **type 3**: peripherals which are both securable and isolable
  – the decprot[] bits define access rules according to *Table 92*
  – failed access are reported with a bus error.

Any resource not listed in *Table 93: Programmable options according to resource type* is accessible by any bus master (for example, non secured).

The debug AXI-AP is non-secure but able to access MCU resources.

Table 90. decprot bits for securable peripherals (type 1) and behavior

decprot[1:0]	Comment
0b00	The peripheral is RW secure. Default after reset.
0b01	The peripheral is write-secure (for example.only write secure are allowed, read can be secure or non-secure).
0b10	Reserved. (Undefined behavior but does not weaken security.)
0b11	The peripheral is not secure (read and write by secure and non-secure means).

Table 91. decprot bits for isolable peripherals (type 2) and behavior

decprot[1:0]	Comment
0b0x	Reserved. (Undefined behavior but does not weaken security).
0b10	Peripheral RW access only by the Cortex-M4 domain. (Cortex-A7 and other bus masters not in the Cortex-M4 domains are denied whether secure or non secure).
0b11	The peripheral is not secure (read and write by secure and non-secure means). Default after reset.

Table 92. decprot bits for securable and Isolable Peripherals (type 3) and behavior

decprot[1:0]	Comment
0b00	The peripheral is RW secure.
0b01	The peripheral is write-secure (for example, only secure writes are allowed, read can be secure or non-secure).
0b10	Peripheral RW access only by Cortex-M4 domain. (Cortex-A7 and other Bus master not in Cortex-M4 domains are denied whether secure or non secure).
0b11	The peripheral is not secure (read and write by secure and non secure means). Default after reset.

Programmable values of decprot[] bits according to resource type are summarized in *Table 93*

Table 93. Programmable options according to resource type

Resource type	0b00 (secured)	0b01 (Write secured)	0b10 (MCU isolation)	0b11 (non-secured)
1: Securable	Prog (default)	Prog	No	Prog
2: MCU isolation	No	No	Prog	Prog (default)
3: Securable and MCU Isolation	Prog	Prog	Prog	Prog (default)

Note: Default after reset indicated by "".*

Bus master and MCU resource isolation:

Bus masters listed in *Table 94: DECPROT assignment* (DMA1, DMA2, OTG, SDMMC3, ETH1) may be assigned to the MCU isolation domain according to their control port decprot[] bit programming value. In this case they have granted access to other MCU isolated resources.

Failed access behaviors:

Whether it is a security or isolation violation, failed writes are ignored and failed reads are returned as zeros. Furthermore a bus error is asserted to either raise an exception with the Cortex CPUs, or to be flagged by an interrupt with DMA IPs.

Note: *The only exception is GPIOZ (Tz-aware IP), which fails silently.*

Any write access to ETZPC locked settings is also silently ignored.

All peripherals concerned by security and MCU resource isolation, and controlled by the ETZPC are listed in *Table 94* with the corresponding IP decprot[] bits.

Table 94. DECPROT assignment

Index	decprot bits	IP	Bus	Default	Bus master	Type	Attributes
0	DECPROT0[1:0]	STGENC	APB5	0b00	-	1	Securable
1	DECPROT0[3:2]	BKPSRAM	AHB5	0b00	-	1	Securable
2	DECPROT0[5:4]	IWDG1	APB5	0b00	-	1	Securable
3	DECPROT0[7:6]	USART1	APB5	0b00	-	1	Securable
4	DECPROT0[9:8]	SPI6	APB5	0b00	-	1	Securable
5	DECPROT0[11:10]	I2C4	APB5	0b00	-	1	Securable
6	DECPROT0[13:12]	-	-	0b00	-	-	Reserved
7	DECPROT0[15:14]	RNG1	AHB5	0b00	-	1	Securable
8	DECPROT0[17:16]	HASH1	AHB5	0b00	-	1	Securable
9	DECPROT0[19:18]	CRYP1	AHB5	0b00	-	1	Securable
10	DECPROT0[21:20]	DDRCTRL	APB4	0b00	-	1	Securable
11	DECPROT0[23:22]	DDRPHYC	APB4	0b00	-	1	Securable

Table 94. DECPROT assignment (continued)

Index	decprot bits	IP	Bus	Default	Bus master	Type	Attributes
12	DECPROT0[25:24]	I2C6	APB5	0b00	-	1	Securable
13	DECPROT0[27:26]	-	-	0b00	-	-	Reserved
14	DECPROT0[29:28]	-	-	0b00	-	-	Reserved
15	DECPROT0[31:30]	-	-	0b00	-	-	Reserved
16	DECPROT1[1:0]	TIM2	APB1	0b11	-	2	Not securable, MCU isolation only
17	DECPROT1[3:2]	TIM3	APB1	0b11	-	2	Not securable, MCU isolation only.
18	DECPROT1[5:4]	TIM4	APB1	0b11	-	2	Not securable, MCU isolation only.
19	DECPROT1[7:6]	TIM5	APB1	0b11	-	2	Not securable, MCU isolation only.
20	DECPROT1[9:8]	TIM6	APB1	0b11	-	2	Not securable, MCU isolation only.
21	DECPROT1[11:10]	TIM7	APB1	0b11	-	2	Not securable, MCU isolation only.
22	DECPROT1[13:12]	TIM12	APB1	0b11	-	2	Not securable, MCU isolation only.
23	DECPROT1[15:14]	TIM13	APB1	0b11	-	2	Not securable, MCU isolation only.
24	DECPROT1[17:16]	TIM14	APB1	0b11	-	2	Not securable, MCU isolation only.
25	DECPROT1[19:18]	LPTIM1	APB1	0b11	-	2	Not securable, MCU isolation only.
26	DECPROT1[21:20]	WWDG1	APB1	0b11	-	2	Not securable, MCU isolation only.
27	DECPROT1[23:22]	SPI2	APB1	0b11	-	2	Not securable, MCU isolation only.
28	DECPROT1[25:24]	SPI3	APB1	0b11	-	2	Not securable, MCU isolation only.
29	DECPROT1[27:26]	SPDIFRX	APB1	0b11	-	2	Not securable, MCU isolation only.
30	DECPROT1[29:28]	USART2	APB1	0b11	-	2	Not securable, MCU isolation only.
31	DECPROT1[31:30]	USART3	APB1	0b11	-	2	Not securable, MCU isolation only.
32	DECPROT2[1:0]	UART4	APB1	0b11	-	2	Not securable, MCU isolation only.
33	DECPROT2[3:2]	UART5	APB1	0b11	-	2	Not securable, MCU isolation only.

Table 94. DECPROT assignment (continued)

Index	decprot bits	IP	Bus	Default	Bus master	Type	Attributes
34	DECPROT2[5:4]	I2C1	APB1	0b11	-	2	Not securable, MCU isolation only
35	DECPROT2[7:6]	I2C2	APB1	0b11	-	2	Not securable, MCU isolation only.
36	DECPROT2[9:8]	I2C3	APB1	0b11	-	2	Not securable, MCU isolation only.
37	DECPROT2[11:10]	I2C5	APB1	0b11	-	2	Not securable, MCU isolation only.
38	DECPROT2[13:12]	CEC	APB1	0b11	-	2	Not securable, MCU isolation only.
39	DECPROT2[15:14]	DAC	APB1	0b11	-	2	Not securable, MCU isolation only.
40	DECPROT2[17:16]	UART7	APB1	0b11	-	2	Not securable, MCU isolation only.
41	DECPROT2[19:18]	UART8	APB1	0b11	-	2	Not securable, MCU isolation only.
42	DECPROT2[21:20]	-		0b11	-	-	Reserved
43	DECPROT2[23:22]	-		0b11	-	-	Reserved
44	DECPROT2[25:24]	MDIOS	APB1	0b11	-	2	Not securable, MCU isolation only.
45	DECPROT2[27:26]	-		0b11	-	-	Reserved
46	DECPROT2[29:28]	-		0b11	-	-	Reserved
47	DECPROT2[31:30]	-		0b11	-	-	Reserved
48	DECPROT3[1:0]	TIM1	APB2	0b11	-	2	Not securable, MCU isolation only.
49	DECPROT3[3:2]	TIM8	APB2	0b11	-	2	Not securable, MCU isolation only.
50	DECPROT3[5:4]	-		0b11	-	-	Reserved
51	DECPROT3[7:6]	USART6	APB2	0b11	-	2	Not securable, MCU isolation only.
52	DECPROT3[9:8]	SPI1	APB2	0b11	-	2	Not securable, MCU isolation only.
53	DECPROT3[11:10]	SPI4	APB2	0b11	-	2	Not securable, MCU isolation only.
54	DECPROT3[13:12]	TIM15	APB2	0b11	-	2	Not securable, MCU isolation only.
55	DECPROT3[15:14]	TIM16	APB2	0b11	-	2	Not securable, MCU isolation only.
56	DECPROT3[17:16]	TIM17	APB2	0b11	-	2	Not securable, MCU isolation only.

Table 94. DECPROT assignment (continued)

Index	decprot bits	IP	Bus	Default	Bus master	Type	Attributes
57	DECPROT3[19:18]	SPI5	APB2	0b11	-	2	Not securable, MCU isolation only
58	DECPROT3[21:20]	SAI1	APB2	0b11	-	2	Not securable, MCU isolation only
59	DECPROT3[23:22]	SAI2	APB2	0b11	-	2	Not securable, MCU isolation only
60	DECPROT3[25:24]	SAI3	APB2	0b11	-	2	Not securable, MCU isolation only
61	DECPROT3[27:26]	DFSDM	APB2	0b11	-	2	Not securable, MCU isolation only
62	DECPROT3[29:28]	TT_FDCAN(1)	APB2	0b11	-	2	Not securable, MCU isolation only
63	DECPROT3[31:30]	-		0b11	-	-	Reserved
64	DECPROT4[1:0]	LPTIM2	APB3	0b11	-	2	Not securable, MCU isolation only
65	DECPROT4[3:2]	LPTIM3	APB3	0b11	-	2	Not securable, MCU isolation only
66	DECPROT4[5:4]	LPTIM4	APB3	0b11	-	2	Not securable, MCU isolation only
67	DECPROT4[7:6]	LPTIM5	APB3	0b11	-	2	Not securable, MCU isolation only
68	DECPROT4[9:8]	SAI4	APB3	0b11	-	2	Not securable, MCU isolation only
69	DECPROT4[11:10]	VREFBUF	APB3	0b11	-	2	Not securable, MCU isolation only
70	DECPROT4[13:12]	DCMI	AHB3	0b11	-	2	Not securable, MCU isolation only
71	DECPROT4[15:14]	CRC2	AHB3	0b11	-	2	Not securable, MCU isolation only
72	DECPROT4[17:16]	ADC	AHB2	0b11	-	2	Not securable, MCU isolation only
73	DECPROT4[19:18]	HASH2	AHB3	0b11	-	2	Not securable, MCU isolation only
74	DECPROT4[21:20]	RNG2	AHB3	0b11	-	2	Not securable, MCU isolation only
75	DECPROT4[23:22]	CRYP2	AHB3	0b11	-	2	Not securable, MCU isolation only
76	DECPROT4[25:24]	-		0b11	-	-	Reserved
77	DECPROT4[27:26]	-		0b11	-	-	Reserved
78	DECPROT4[29:28]	-		0b11	-	-	Reserved

Table 94. DECPROT assignment (continued)

Index	decprot bits	IP	Bus	Default	Bus master	Type	Attributes
79	DECPROT4[31:30]	-		0b11	-	-	Reserved
80	DECPROT5[1:0]	SRAM1	MLAHB	0b11	-	3	Securable and MCU isolation support
81	DECPROT5[3:2]	SRAM2	MLAHB	0b11	-	3	Securable and MCU isolation support
82	DECPROT5[5:4]	SRAM3	MLAHB	0b11	-	3	Securable and MCU isolation support
83	DECPROT5[7:6]	SRAM4	MLAHB	0b11	-	3	Securable and MCU isolation support
84	DECPROT5[9:8]	RETRAM	MLAHB	0b11	-	3	Securable and MCU isolation support
85	DECPROT5[11:10]	OTG(2)	AHB2	0b11	yes	2	Not securable, MCU isolation only
86	DECPROT5[13:12]	SDMMC3	AHB2	0b11	yes	2	Not securable, MCU isolation only
87	DECPROT5[15:14]	DLYBSD3	AHB2	0b11	-	2	Not securable, MCU isolation only
88	DECPROT5[17:16]	DMA1	AHB2	0b11	yes	2	Not securable, MCU isolation only
89	DECPROT5[19:18]	DMA2	AHB2	0b11	yes	2	Not securable, MCU isolation only
90	DECPROT5[21:20]	DMAMUX	AHB2	0b11	-	2	Not securable, MCU isolation only
91	DECPROT5[23:22]	FMC	AXIM	0b11	-	2	Not securable, MCU isolation only
92	DECPROT5[25:24]	QSPI	AXIM	0b11	-	2	Not securable, MCU isolation only
93	DECPROT5[27:26]	DLYBQ	AHB6	0b11	-	2	Not securable, MCU isolation only
94	DECPROT5[29:28]	ETH1	AHB6	0b11	yes	2	Not securable, MCU isolation only
95	DECPROT5[31:30]	-		0b11	-	-	Reserved

1. FDCAN use one decprot[] for several APB regions
2. USBPHYC associated to OTG cannot be isolated

15.7 ETZPC registers

The ETZPC registers are accessed only by word (32-bit).

ETZPC registers are write-secure (that is, read accesses are allowed by secure and non-secure domains, write accesses are allowed by the secure domain only).

The ETZPC_DECPROTx registers are used to control security and resource isolation for all peripherals listed in *Table 94: DECPROT assignment* according to the index (from 0 to 95).

Caution: The ETZPC is programmed during early boot, it should not be changed afterwards, lock bits can enforce this; propagation of the ETZPC decprot[] bits takes some delay, two APB cycles must be observed before security rule is applied.

15.7.1 ETZPC ROM secure size definition (ETZPC_TZMA0_SIZE)

Address offset: 0x000

Reset value: 0x0000 03FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	R0SIZE[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: To lock r0size settings of the TZMA

0: r0size[] can be written

1: r0size[] cannot be modified until the next reset, therefore the write to r0size is silently ignored (for example no APB SLVERR).

Bits 30:10 Reserved, must be kept at reset value.

Bits 9:0 **R0SIZE[9:0]**

Defines the size of the secure ROM at the lower values, and non-secure ROM at the higher values:

0x000: all ROM is non-secure

0x001: 4 Kbyte ROM secure

0x002: 8 Kbyte ROM secure

....

0x020 to 0x3FF: 128 Kbyte ROM secure (for example all ROM secure)

15.7.2 ETZPC RAM secure size definition (ETZPC_TZMA1_SIZE)

Address offset: 0x004

Reset value: 0x0000 03FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	R0SIZE[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LOCK**: To lock r0size settings of the TZMA

0: r0size[] can be written

1: r0size[] can not be modified until next reset, therefore the write to r0size is silently ignored (for example APB SLVERR)

Bits 30:10 Reserved, must be kept at reset value.

Bits 9:0 **R0SIZE[9:0]:** Defines the size of the secure RAM at the lower values, and non-secure RAM at the higher values:

0x000: all RAM is non-secure

0x001: 4 Kbyte RAM secure

0x002: 8 Kbyte RAM secure

....

0x040 to 0x3FF: 256 Kbyte RAM secure (for example all RAM secure)

15.7.3 ETZPC securable peripheral control register x (ETZPC_DECPROTx)

Address offset: 0x010 + 0x004 * (x- 0) (x = 0 to 5)

Reset value: 0x0000 0000, 0xFFFF FFFF, 0xFFFF FFFF, 0xFFFF FFFF, 0xFFFF FFFF, 0xFFFF FFFF

Register reset values according to values in *Table 94: DECPROT assignment* column *Default*, see also register map.

Define security and isolation for peripheral at index (16 * x + y) in *Table 94* according to ETZPC_DECPROTx with (x = 0 to 5) and (y= 0 to 15) according to DECPROTy[1:0]:

- 00: read and write secure
- 01: read non-secure and write secure
- 10: non-secure but MCU isolated
- 11: non-secure.

Permissible values of DECPROTy[1:0] are according to *Table 94* column *Attributes*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DECPROT15[1:0]		DECPROT14[1:0]		DECPROT13[1:0]		DECPROT12[1:0]		DECPROT11[1:0]		DECPROT10[1:0]		DECPROT9[1:0]		DECPROT8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DECPROT7[1:0]		DECPROT6[1:0]		DECPROT5[1:0]		DECPROT4[1:0]		DECPROT3[1:0]		DECPROT2[1:0]		DECPROT1[1:0]		DECPROT0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DECPROT15[1:0]**: Settings for peripheral with index (16 * x + 15) in *Table 94*

Bits 29:28 **DECPROT14[1:0]**: Settings for peripheral with index (16 * x + 14) in *Table 94*

Bits 27:26 **DECPROT13[1:0]:** Settings for peripheral with index (16 * x + 13) in *Table 94*

Bits 25:24 **DECPROT12[1:0]:** Settings for peripheral with index (16 * x + 12) in *Table 94*

Bits 23:22 **DECPROT11[1:0]:** Settings for peripheral with index (16 * x + 11) in *Table 94*

Bits 21:20 **DECPROT10[1:0]:** Settings for peripheral with index (16 * x + 10) in *Table 94*.

Bits 19:18 **DECPROT9[1:0]:** Settings for peripheral with index (16 * x + 9) in *Table 94*.

Bits 17:16 **DECPROT8[1:0]:** Settings for peripheral with index (16 * x + 8) in *Table 94*.

Bits 15:14 **DECPROT7[1:0]:** Settings for peripheral with index (16 * x + 7) in *Table 94*.

Bits 13:12 **DECPROT6[1:0]:** Settings for peripheral with index (16 * x + 6) in *Table 94*.

Bits 11:10 **DECPROT5[1:0]:** Settings for peripheral with index (16 * x + 5) in *Table 94*.

Bits 9:8 **DECPROT4[1:0]:** Settings for peripheral with index (16 * x + 4) in *Table 94*.

Bits 7:6 **DECPROT3[1:0]:** Settings for peripheral with index (16 * x + 3) in *Table 94*.

Bits 5:4 **DECPROT2[1:0]:** Settings for peripheral with index (16 * x + 2) in *Table 94*.

Bits 3:2 **DECPROT1[1:0]:** Settings for peripheral with index (16 * x + 1) in *Table 94*.

Bits 1:0 **DECPROT0[1:0]:** Settings for peripheral with index (16 * x + 0) in *Table 94*.

15.7.4 ETZPC decprot lock x register (ETZPC_DECPROT_LOCKx)

Address offset: 0x030 + 0x004 * (x-0) (x = 0 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK31	LOCK30	LOCK29	LOCK28	LOCK27	LOCK26	LOCK25	LOCK24	LOCK23	LOCK22	LOCK21	LOCK20	LOCK19	LOCK18	LOCK17	LOCK16
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bit 31 **LOCK31:** Lock of security for peripheral with index (32 * x + 31) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 30 **LOCK30**: Lock of security for peripheral with index (32 * x + 30) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 29 **LOCK29**: Lock of security for peripheral with index (32 * x + 29) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 28 **LOCK28**: Lock of security for peripheral with index (32 * x + 28) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 27 **LOCK27**: Lock of security for peripheral with index (32 * x + 27) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 26 **LOCK26**: Lock of security for peripheral with index (32 * x + 26) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 25 **LOCK25**: Lock of security for peripheral with index (32 * x + 25) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 24 **LOCK24**: Lock of security for peripheral with index (32 * x + 24) as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 23 **LOCK23**: Lock of security for peripheral with index (32 * x + 23) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 22 **LOCK22**: Lock of security for peripheral with index (32 * x + 22) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 21 **LOCK21**: Lock of security for peripheral with index (32 * x + 21) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 20 **LOCK20**: Lock of security for peripheral with index (32 * x + 20) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 19 **LOCK19**: Lock of security for peripheral with index (32 * x + 19) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 18 **LOCK18**: Lock of security for peripheral with index (32 * x + 18) as defined by *Table 94*.
0: not locked
1: decprot[] bits are locked until next reset

Bit 17 **LOCK17**: Lock of security for peripheral with index (32 * x + 17) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 16 **LOCK16**: Lock of security for peripheral with index (32 * x + 16) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 15 **LOCK15:** Lock of security for peripheral with index (32 * x + 15) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 14 **LOCK14**: Lock of security for peripheral with index (32 * x + 14) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 13 **LOCK13**: Lock of security for peripheral with index (32 * x + 13) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 12 **LOCK12**: Lock of security for peripheral with index (32 * x + 12) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 11 **LOCK11**: Lock of security for peripheral with index (32 * x + 11) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 10 **LOCK10**: Lock of security for peripheral with index (32 * x + 10) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 9 **LOCK9**: Lock of security for peripheral with index (32 * x + 9) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 8 **LOCK8**: Lock of security for peripheral with index (32 * x + 8) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 7 **LOCK7**: Lock of security for peripheral with index (32 * x + 7) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 6 **LOCK6**: Lock of security for peripheral with index (32 * x + 6) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 5 **LOCK5**: Lock of security for peripheral with index (32 * x + 5) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 4 **LOCK4**: Lock of security for peripheral with index (32 * x + 4) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 3 **LOCK3**: Lock of security for peripheral with index (32 * x + 3) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 2 **LOCK2**: Lock of security for peripheral with index (32 * x + 2) as defined by *Table 94*.
0: not locked
1: decprot[] bits are locked until next reset

Bit 1 **LOCK1**: Lock of security for peripheral with index (32 * x + 1) as defined by *Table 94*.
0: not locked
1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

Bit 0 **LOCK0**: Lock of security for peripheral with index 32 * x as defined by *Table 94*.

0: not locked

1: peripheral decprot[] bits are locked until next reset, therefore the write to decprot[] is silently ignored (for example no APB SLVERR)

15.7.5 ETZPC HW configuration register (ETZPC_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 6002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHUNKS1N4[7:0]								NUM_AHB_SEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NUM_PER_SEC[7:0]								NUM_TZMA[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **CHUNKS1N4[7:0]**: TZMA size steps (0: 4 Kbytes, 1: 1 Kbyte)

Bits 23:16 **NUM_AHB_SEC[7:0]**: Number of securable AHB masters

Bits 15:8 **NUM_PER_SEC[7:0]**: Number of securable peripherals

Bits 7:0 **NUM_TZMA[7:0]**: Number of TZMA instances

15.7.6 ETZPC version register (ETZPC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision

This field returns the major revision of the ETZPC

Bits 3:0 **MINREV[3:0]:** Minor revision

This field returns the minor revision of the ETZPC

15.7.7 ETZPC identification register (ETZPC_IDR)

Address offset: 0x3F8

Reset value: 0x0010 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code.
This field returns the Identification code of ETZPC

15.7.8 ETZPC size identification register (ETZPC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size and ID.
This field returns the size and ID of ETZPC

15.7.9 ETZPC register map

The table below provides the ETZPC register map and reset values.

Table 95. ETZPC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ETZPC_TZMA0_SIZE	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0SIZE									
	Reset value	0																						1	1	1	1	1	1	1	1	1	1
0x004	ETZPC_TZMA1_SIZE	LOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	R0SIZE									
	Reset value	0																						1	1	1	1	1	1	1	1	1	1
0x010	ETZPC_DECPROT0	DECPROT15.		DECPROT14		DECPROT13		DECPROT12		DECPROT11		DECPROT10		DECPROT9		DECPROT8		DECPROT7		DECPROT6		DECPROT5		DECPROT4		DECPROT3		DECPROT2		DECPROT1		DECPROT0	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ETZPC_DECPROT1	DECPROT15		DECPROT14		DECPROT13		DECPROT12		DECPROT11		DECPROT10		DECPROT9		DECPROT8		DECPROT7		DECPROT6		DECPROT5		DECPROT4		DECPROT3		DECPROT2		DECPROT1		DECPROT0	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x018	ETZPC_DECPROT2	DECPROT15.		DECPROT14		DECPROT13		DECPROT12		DECPROT11		DECPROT10		DECPROT9		DECPROT8		DECPROT7		DECPROT6		DECPROT5		DECPROT4		DECPROT3		DECPROT2		DECPROT1		DECPROT0	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x01C	ETZPC_DECPROT3	DECPROT15.		DECPROT14		DECPROT13		DECPROT12		DECPROT11		DECPROT10		DECPROT9		DECPROT8		DECPROT7		DECPROT6		DECPROT5		DECPROT4		DECPROT3		DECPROT2		DECPROT1		DECPROT0	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x020	ETZPC_DECPROT4	DECPROT15.		DECPROT14		DECPROT13		DECPROT12		DECPROT11		DECPROT10		DECPROT9		DECPROT8		DECPROT7		DECPROT6		DECPROT5		DECPROT4		DECPROT3		DECPROT2		DECPROT1		DECPROT0	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x024	ETZPC_DECPROT5	DECPROT15.		DECPROT14		DECPROT13		DECPROT12		DECPROT11		DECPROT10		DECPROT9		DECPROT8		DECPROT7		DECPROT6		DECPROT5		DECPROT4		DECPROT3		DECPROT2		DECPROT1		DECPROT0	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	ETZPC_DECPROT_LOCK0	LOCK31	LOCK30	LOCK29	LOCK28	LOCK27	LOCK26	LOCK25	LOCK24	LOCK23	LOCK22	LOCK21	LOCK20	LOCK19	LOCK18	LOCK17	LOCK16	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETZPC_DECPROT_LOCK1	LOCK31	LOCK30	LOCK29	LOCK28	LOCK27	LOCK26	LOCK25	LOCK24	LOCK23	LOCK22	LOCK21	LOCK20	LOCK19	LOCK18	LOCK17	LOCK16	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	ETZPC_DECPROT_LOCK3	LOCK31	LOCK30	LOCK29	LOCK28	LOCK27	LOCK26	LOCK25	LOCK24	LOCK23	LOCK22	LOCK21	LOCK20	LOCK19	LOCK18	LOCK17	LOCK16	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 95. ETZPC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F0	ETZPC_HWCFGR	CHUNKS1N4								NUM_AHB_SEC[7:0]								NUM_PER_SEC[7:0]								NUM_TZMA[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0
0x3F4	ETZPC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res.	Res	Res	Res	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	0	0	0	0	0
0x3F8	ETZPC_IDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
0x3FC	ETZPC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

16 Block interconnects

16.1 Peripheral interconnects

16.1.1 Introduction

Several peripherals have direct interconnections between them. This allows autonomous communication and synchronization between peripherals, saving processor resources and reducing power supply consumption.

In addition, these hardware connections remove software latency and allow the design of a predictable system.

This also saves pins and GPIOs.

Depending on the peripheral, these interconnections can operate in Run, CSleep or Stop modes.

16.1.2 Connection overview

There are several types of connections, as listed below.

Asynchronous connections (A)

The source output signal is sampled by the destination clock, leading to the possible introduction of jitter in the latency between the source output event and the destination event detection.

Synchronous connections (S)

Both source and destination are synchronous (they run on the same clock), and the latency from the source to the destination is deterministic. No jitter is introduced.

Break/fault connection for TIM outputs (B)

The source output signal disables the timer outputs through a pure combinatorial logic path, without any latency.

Table 96. Peripherals interconnect matrix

			Destination																								
Domain	Bus	Domain	MCU																								MPU
		Bus	AHB2	APB1							APB2										APB3						AXIM
		Peripheral	ADC1/ADC2	DAC1/DAC2	LPTIM1	TIM12	TIM2	TIM3	TIM4	TIM5	FDCAN	DFSDM1	TIM1	TIM15	TIM16	TIM17	TIM8	SAI1	SAI2	SAI3	SAI4	LPTIM2	LPTIM3	LPTIM4	LPTIM5	DTS	ETH1
-	-	AFMUX			A	A	A	A	A	A			A/B	A/B	A/B	A/B	A/B					A					
MCU	APB1	LPTIM1	A	S								A														A	
		SPDIFRX														A											
		TIM2	A/S	S				S	S		A		S				S										A
		TIM3	A/S				S		S	S	A	S	S	S													A
		TIM4	A/S	S		S	S	S		S		S	S				S										
		TIM5		S		S											S										
		TIM6	S	S								S															
		TIM7		S								S															
		TIM13				S																					
		TIM14				S																					

Table 96. Peripherals interconnect matrix (continued)

			Destination																								
Domain	Bus	Domain	MCU																								MPU
		Bus	AHB2	APB1							APB2							APB3									AXIM
		Peripheral	ADC1/ADC2	DAC1/DAC2	LPTIM1	TIM12	TIM2	TIM3	TIM4	TIM5	FDCAN	DFSDM1	TIM1	TIM15	TIM16	TIM17	TIM8	SAI1	SAI2	SAI3	SAI4	LPTIM2	LPTIM3	LPTIM4	LPTIM5	DTS	ETH1
MCU	APB2	FDCAN								A/S																	A
		DFSDM1											B	B	B	B	B										
		TIM1	A/S	S			S	S	S	S		S		S			S										
		TIM8	A/S	S			S		S	S		S															
		TIM15	S	S				S					S														
		TIM16										S		S													
		TIM17												S													
		SAI1					A												S	S	A		A				
		SAI2								A								S		S	A			A			
		SAI3																S	S		A						
MCU	APB3	SAI4																A	A	A			S		S		
		LPTIM2	A	A								A											S	S	S	S	
		LPTIM3	A									A												S	S	S	
		LPTIM4																					S		S		
		LPTIM5																					S	S			

Table 96. Peripherals interconnect matrix (continued)

			Destination																								
Domain	Bus	Domain	MCU																								MPU
		Bus	AHB2	APB1							APB2										APB3						AXIM
		Peripheral	ADC1/ADC2	DAC1/DAC2	LPTIM1	TIM12	TIM2	TIM3	TIM4	TIM5	FDCAN	DFSDM1	TIM1	TIM15	TIM16	TIM17	TIM8	SAI1	SAI2	SAI3	SAI4	LPTIM2	LPTIM3	LPTIM4	LPTIM5	DTS	ETH1
MCU	AHB2	ADC1										A	A				A										
		ADC2										A	A				A										
	AHB4	RCC				A	A							A	A	A											
		EXTI	A	A								A															
	MLAHB	OTG								S																	
MPU	AXIM	ETH1					A	A			A																
		USBH								A																	
	APB5	RTC			A										A							A					
		TAMP			A																	A					

Table 97. Peripherals interconnect matrix details

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB2	TIM15	TRGO	ITR0	TIM1	APB2	MCU	S
	APB1	TIM2	TRGO	ITR1				S
		TIM3	TRGO	ITR2				S
		TIM4	TRGO	ITR3				S
	-	AFMUX	TIM1_ETR	ETR0				A
			NC	ETR1				-
			NC	ETR2				-
	AHB2	ADC1	AWD1	ETR3				A
			AWD2	ETR4				A
			AWD3	ETR5				A
		ADC2	AWD1	ETR6				A
			AWD2	ETR7				A
			AWD3	ETR8				A
	--	AFMUX	TIM1_CH1	TI1_0				A
			TIM1_CH2	TI2_0				A
			TIM1_CH3	TI3_0				A
			TIM1_CH4	TI4_0				A
			TIM1_BKIN	BRK_0				B
			NC	BRK_1				-
			NC	BRK_2				-
			NC	BRK_3				-
			NC	BRK_4				-
			NC	BRK_5				-

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	-	AFMUX	NC	BRK_6	TIM1	APB2	MCU	-
			NC	BRK_7				-
	APB2	DFSDM1	BRK0	BRK_8				B
	-	AFMUX	TIM1_BKIN2	BRK2_0				B
			NC	BRK2_1				-
			NC	BRK2_2				-
			NC	BRK2_3				-
			NC	BRK2_4				-
			NC	BRK2_5				-
			NC	BRK2_6				-
			NC	BRK2_7				-
	APB2	DFSDM1	BRK1	BRK2_8				B
		TIM1	TRGO	ITR0	TIM2	APB1	MCU	S
		TIM8	TRGO	ITR1				S
	APB1	TIM3	TRGO	ITR2				S
		TIM4	TRGO	ITR3				S
	-	AFMUX	TIM2_ETR	ETR0				A
			NC	ETR1				-
			NC	ETR2				-
	AHB4	RCC	LSE	ETR3				A
	APB2	SAI1	FS_A	ETR4				A
			FS_B	ETR5				A
MPU	AXIM	ETH1	PPS	ETR6				A

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	-AXIM	AFMUX	TIM2_CH1	TI1_0	TIM2	APB1	MCU	A
			TIM2_CH2	TI2_0				A
			TIM2_CH3	TI3_0				A
			TIM2_CH4	TI4_0				A
	APB2	TIM1	TRGO	ITR0	TIM3	APB1	MCU	S
	APB1	TIM2	TRGO	ITR1				S
	APB2	TIM15	TRGO	ITR2				S
	APB1	TIM4	TRGO	ITR3				S
	-	AFMUX	TIM3_ETR	ETR0				A
			NC	ETR1				-
			NC	ETR2				-
			NC	ETR3		APB1	MCU	-
			NC	ETR4		APB1	MCU	-
			NC	ETR5				-
MPU	AXIM	ETH1	PPS	ETR6				A
MCU	-	AFMUX	TIM3_CH1	TI1_0				A
			TIM3_CH2	TI2_0				A
			TIM3_CH3	TI3_0				A
			TIM3_CH4	TI4_0				A
	APB2	TIM1	TRGO	ITR0	TIM4	APB1	MCU	S
	APB1	TIM2	TRGO	ITR1				S
		TIM3	TRGO	ITR2				S
	APB2	TIM8	TRGO	ITR3				S

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	-	AFMUX	TIM4_ETR	ETR0	TIM4	APB1	MCU	A
			TIM4_CH1	TI1_0				A
			TIM4_CH2	TI2_0				A
			TIM4_CH3	TI3_0				A
			TIM4_CH4	TI4_0				A
	APB2	TIM1	TRGO	ITR0	TIM5	APB1	MCU	S
		TIM8	TRGO	ITR1				S
	APB1	TIM3	TRGO	ITR2				S
		TIM4	TRGO	ITR3				S
	APB2	FDCAN	SOC	ITR6				S
			NC	ITR7				-
	MLAHB	OTG	SOF	ITR8				S
	-	AFMUX	TIM5_ETR	ETR0				A
	APB2	SAI2	FS_A	ETR1				A
			FS_B	ETR2				A
MPU	AXIM	USBH	SOF	ETR3				A
MCU	-	AFMUX	TIM5_CH1	TI1_0				A
	APB2	FDCAN	TMP	TI1_1				A
			RTP	TI1_2				A
	-	AFMUX	TIM5_CH2	TI2_0				A
			TIM5_CH3	TI3_0				A
			TIM5_CH4	TI4_0				A
	APB2	TIM1	TRGO	ITR0	TIM8	APB2	MCU	S

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB1	TIM2	TRGO	ITR1	TIM8	APB2	MCU	S
		TIM4	TRGO	ITR2				S
		TIM5	TRGO	ITR3				S
	-	AFMUX	TIM8_ETR	ETR0				A
			NC	ETR1				-
			NC	ETR2				-
	AHB2	ADC1	AWD1	ETR3				A
			AWD2	ETR4				A
			AWD3	ETR5				A
		ADC2	AWD1	ETR6				A
			AWD2	ETR7				A
			AWD3	ETR8				A
	-	AFMUX	TIM8_CH1	TI1_0				A
			TIM8_CH2	TI2_0				A
			TIM8_CH3	TI3_0				A
			TIM8_CH4	TI4_0				A
			TIM8_BKIN	BRK_0				B
			NC	BRK_1				-
			NC	BRK_2				-
			NC	BRK_3				-
			NC	BRK_4				-
			NC	BRK_5				-
			NC	BRK_6				-

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	**Bus**	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**	**Domain**	
MCU	-	AFMUX	NC	BRK_7	TIM8	APB2	MCU	-
	APB2	DFSDM1	BRK2	BRK_8				B
	-	AFMUX	TIM8_BKIN2	BRK2_0				B
			NC	BRK2_1				-
			NC	BRK2_2				-
			NC	BRK2_3				-
			NC	BRK2_4				-
			NC	BRK2_5				-
			NC	BRK2_6				-
			NC	BRK2_7				-
	APB2	DFSDM1	BRK3	BRK2_8				B
	APB1	TIM4	TRGO	ITR0	TIM12	APB1	MCU	S
		TIM5	TRGO	ITR1				S
		TIM13	OC1	ITR2				S
		TIM14	OC1	ITR3				S
	-	AFMUX	TIM12_CH1	TI1_0				A
	AHB4	RCC	hsi_cal_ck	TI1_1				A
			csi_cal_ck	TI1_2				A
	-	AFMUX	TIM12_CH2	TI2_0				A
			TIM13_CH1	TI1_0	TIM13	APB1	MCU	A
			TIM14_CH1	TI1_0	TIM14	APB1	MCU	A
	APB2	TIM1	TRGO	ITR0	TIM15	APB2	MCU	S
	APB1	TIM3	TRGO	ITR1				S
	APB2	TIM16	OC1	ITR2				S
		TIM17	OC1	ITR3				S

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	-	AFMUX	TIM15_CH1	TI1_0	TIM15	APB2	MCU	A
			TIM2_CH1	TI1_1				A
			TIM3_CH1	TI1_2				A
			TIM4_CH1	TI1_3				A
	AHB4	RCC	LSE	TI1_4				A
			CSI	TI1_5				A
			MCO2	TI1_6				A
			hsi_cal_ck	TI1_7				A
			csi_cal_ck	TI1_8				A
	-	AFMUX	TIM15_CH2	TI2_0				A
			TIM2_CH2	TI2_1				A
			TIM3_CH2	TI2_2				A
			TIM4_CH2	TI2_3				A
			TIM15_BKIN	BRK_0				B
			NC	BRK_1				-
			NC	BRK_2				-
			NC	BRK_3				-
			NC	BRK_4				-
			NC	BRK_5				-
			NC	BRK_6				-
			NC	BRK_7				-
	APB2	DFSDM1	BRK0	BRK_8				B
	-	AFMUX	TIM16_CH1	TI1_0	TIM16	APB2	MCU	A
	AHB4	RCC	LSI	TI1_1				A
			LSE	TI1_2				A

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MPU	APB5	RTC	WKUP	TI1_3	TIM16	APB2	MCU	A
MCU	-	AFMUX	TIM16_BKIN	BRK_0				B
			NC	BRK_1				-
			NC	BRK_2				-
			NC	BRK_3				-
			NC	BRK_4				-
			NC	BRK_5				-
			NC	BRK_6				-
			NC	BRK_7				-
	APB2	DFSDM1	BRK1	BRK_8				B
	-	AFMUX	TIM17_CH1	TI1_0	TIM17	APB2	MCU	A
	APB1	SPDIFRX	FS	TI1_1				A
	AHB4	RCC	HSE_RTC	TI1_2				A
			MCO1	TI1_3				A
	-	AFMUX	TIM17_BKIN	BRK_0				B
			NC	BRK_1				-
			NC	BRK_2				-
			NC	BRK_3				-
			NC	BRK_4				-
			NC	BRK_5				-
			NC	BRK_6				-
			NC	BRK_7				-
	APB2	DFSDM1	BRK2	BRK_8				B

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	-	AFMUX	LPTIM1_ETR	ETR0	LPTIM1	APB1	MCU	A
MPU	APB5	RTC	ALA	ETR1				A
			ALB	ETR2				A
		TAMP	TAMP1	ETR3				A
			TAMP2	ETR4				A
			TAMP3	ETR5				A
			NC	ETR6				-
			NC	ETR7				-
MCU	-	AFMUX	LPTIM1_IN1	IN1_0				A
			LPTIM1_IN2	IN2_0				A
			LPTIM2_ETR	ETR0	LPTIM2	APB3	MCU	A
MPU	APB5	RTC	ALA	ETR1				A
			ALB	ETR2				A
		TAMP	TAMP1	ETR3				A
			TAMP2	ETR4				A
			TAMP3	ETR5				A
			NC	ETR6				-
			NC	ETR7				-
MCU	-	AFMUX	LPTIM2_IN1	IN1_0				A
			LPTIM2_IN2	IN2_0				A
	APB3	LPTIM2	OUT	ETR0	LPTIM3	APB3	MCU	S
			NC	ETR1				-
		LPTIM4	OUT	ETR2				S

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB3	LPTIM5	OUT	ETR3	LPTIM3	APB3	MCU	S
	APB2	SAI1	FS_A	ETR4				A
			FS_B	ETR5				A
			NC	IN1_0				-
	APB3	SAI4	FS_A	IN1_1				S
			FS_B	IN1_2				S
		LPTIM2	OUT	ETR0	LPTIM4	APB3	MCU	S
		LPTIM3	OUT	ETR1				S
			NC	ETR2				-
		LPTIM5	OUT	ETR3				S
	APB2	SAI2	FS_A	ETR4				A
			FS_B	ETR5				A
	APB3	LPTIM2	OUT	ETR0	LPTIM5	APB3	MCU	S
		LPTIM3	OUT	ETR1				S
		LPTIM4	OUT	ETR2				S
		SAI4	FS_A	ETR3				S
			FS_B	ETR4				S
			NC	ETR5				-
	APB1	DAC1/DAC 2	SWTRIG	ITR0	DAC1/DAC2	APB1	MCU	S
	APB2	TIM1	TRGO	ITR1				S
	APB1	TIM2	TRGO	ITR2				S
		TIM4	TRGO	ITR3				S
		TIM5	TRGO	ITR4				S
MCU	APB1	TIM6	TRGO	ITR5	DAC1/DAC2	APB1	MCU	S
		TIM7	TRGO	ITR6				S
	APB2	TIM8	TRGO	ITR7				S
		TIM15	TRGO	ITR8				S
			NC	ITR9				-
			NC	ITR10				-
	APB1	LPTIM1	OUT	ITR11				S
	APB3	LPTIM2	OUT	ITR12				A
MCU	AHB4	EXTI	EXTI9	ITR13				A

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB2	TIM1	TRGO	TRG0	DFSDM1	APB2	MCU	S
			TRGO2	TRG1				S
		TIM8	TRGO	TRG2				S
			TRGO2	TRG3				S
	APB1	TIM3	TRGO	TRG4				S
		TIM4	TRGO	TRG5				S
	APB2	TIM16	OC1	TRG6				S
	APB1	TIM6	TRGO	TRG7				S
		TIM7	TRGO	TRG8				S
	-	-	NC	TRG9				-
	-	-	NC	TRG10				-
MCU	AHB4	EXTI	EXTI11	TRG24				A
		-	EXTI15	TRG25				A
MCU	APB1	LPTIM1	OUT	TRG26				A

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB3	LPTIM2	OUT	TRG27	DFSDM1	APB2	MCU	A
		LPTIM3	OUT	TRG28				A
	AHB2	ADC1	DATA[15:0]	DAT_ADC1[15:0]				A
		ADC2	DATA[15:0]	DAT_ADC2[15:0]				A
	APB2	TIM1	CC1	EXT0	ADC1/ADC2	AHB2	MCU	A
			CC2	EXT1				A
			CC3	EXT2				A
	APB1	TIM2	CC2	EXT3				A
		TIM3	TRGO	EXT4				S
		TIM4	CC4	EXT5				A
	AHB4	EXTI	EXTI11	EXT6				A
	APB2	TIM8	TRGO	EXT7				S
			TRGO2	EXT8				S
		TIM1	TRGO	EXT9				S
			TRGO2	EXT10				S
	APB1	TIM2	TRGO	EXT11				S
		TIM4	TRGO	EXT12				S
		TIM6	TRGO	EXT13				S
	APB2	TIM15	TRGO	EXT14				S
	APB1	TIM3	CC4	EXT15				A
			NC	EXT16				-
			NC	EXT17				-
		LPTIM1	OUT	EXT18				A
	APB3	LPTIM2	OUT	EXT19				A
		LPTIM3	OUT	EXT20				A

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB2	TIM1	TRGO	JEXT0	ADC1/ADC2	AHB2	MCU	S
			CC4	JEXT1				A
	APB1	TIM2	TRGO	JEXT2				S
			CC1	JEXT3				A
		TIM3	CC4	JEXT4				A
		TIM4	TRGO	JEXT5				S
MCU	AHB4	EXTI	EXTI15	JEXT6				A
MCU	APB2	TIM8	CC4	JEXT7				A
		TIM1	TRGO2	JEXT8				S
		TIM8	TRGO	JEXT9				S
			TRGO2	JEXT10				S
	APB1	TIM3	CC3	JEXT11				A
			TRGO	JEXT12				S
			CC1	JEXT13				A
		TIM6	TRGO	JEXT14				S
	APB2	TIM15	TRGO	JEXT15				S
			NC	JEXT16				-
			NC	JEXT17				-
	APB1	LPTIM1	OUT	JEXT18				A
	APB3	LPTIM2	OUT	JEXT19				A
		LPTIM3	OUT	JEXT20				A
	APB1	LPTIM1	OUT	TRG1	DTS	APB3	MCU	A
	APB3	LPTIM2	OUT	TRG2				S
MCU	APB3	LPTIM3	OUT	TRG3	DTS	APB3	MCU	S

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB1	TIM2	TRGO	SWT0	FDCAN	APB2	MCU	A
		TIM3	TRGO	SWT1				A
MPU	AXIM	ETH1	PPS	SWT2				A
MCU	APB1	TIM2	TRGO	EVT0				A
		TIM3	TRGO	EVT1				A
MPU	AXIM	ETH1	PPS	EVT2				A
MCU	APB1	TIM3	CNT[15:0]	EXT_TS[15:0]				A
		TIM2	TRGO	PTP0	ETH1	AXIM	MPU	A
		TIM3	TRGO	PTP1				A
			NC	PTP2				-
	APB2	FDCAN	TMP	PTP3				A
			NC	SYNCIN0_FS	SAI1	APB2	MCU	-
			NC	SYNCIN0_SCK				-
		SAI2	SYNCOUT_FS	SYNCIN1_FS				S
			SYNCOUT_SCK	SYNCIN1_SCK				S
		SAI3	SYNCOUT_FS	SYNCIN2_FS				S
			SYNCOUT_SCK	SYNCIN2_SCK				S
	APB3	SAI4	SYNCOUT_FS	SYNCIN3_FS				A
			SYNCOUT_SCK	SYNCIN3_SCK				A
	APB2	SAI1	SYNCOUT_FS	SYNCIN0_FS	SAI2	APB2	MCU	S
			SYNCOUT_SCK	SYNCIN0_SCK				S

Table 97. Peripherals interconnect matrix details (continued)

Source				Destination				Type
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MCU	APB2	SAI1	NC	SYNCIN1_FS	SAI2	APB2	MCU	-
			NC	SYNCIN1_SCK				-
		SAI3	SYNCOUT_FS	SYNCIN2_FS				S
			SYNCOUT_SCK	SYNCIN2_SCK				S
	APB3	SAI4	SYNCOUT_FS	SYNCIN3_FS				A
			SYNCOUT_SCK	SYNCIN3_SCK				A
	APB2	SAI1	SYNCOUT_FS	SYNCIN0_FS	SAI3	APB2	MCU	S
			SYNCOUT_SCK	SYNCIN0_SCK				S
		SAI2	SYNCOUT_FS	SYNCIN1_FS				S
			SYNCOUT_SCK	SYNCIN1_SCK				S
			NC	SYNCIN2_FS				-
			NC	SYNCIN2_SCK				-
	APB3	SAI4	SYNCOUT_FS	SYNCIN3_FS				A
			SYNCOUT_SCK	SYNCIN3_SCK				A
	APB2	SAI1	SYNCOUT_FS	SYNCIN0_FS	SAI4	APB3	MCU	A
			SYNCOUT_SCK	SYNCIN0_SCK				A
		SAI2	SYNCOUT_FS	SYNCIN1_FS				A
			SYNCOUT_SCK	SYNCIN1_SCK				A
		SAI3	SYNCOUT_FS	SYNCIN2_FS				A
			SYNCOUT_SCK	SYNCIN2_SCK				A
			NC	SYNCIN3_FS				-
			NC	SYNCIN3_SCK				-

16.2 MDMA

Table 98. MDMA connections

Source				Destination				Comment
Domain	**Bus**	**Peripheral**	**Signal**	**Signal**	**Peripheral**	**Bus**	**Domain**	
MCU	MLAHB	DMA1	tcif0	str0	MDMA	AXIM	MPU	DMA1 stream 0 transfer complete
			tcif1	str1				DMA1 stream 1 transfer complete
			tcif2	str2				DMA1 stream 2 transfer complete
			tcif3	str3				DMA1 stream 3 transfer complete
			tcif4	str4				DMA1 stream 4 transfer complete
			tcif5	str5				DMA1 stream 5 transfer complete
			tcif6	str6				DMA1 stream 6 transfer complete
			tcif7	str7				DMA1 stream 7 transfer complete
MCU	MLAHB	DMA2	tcif0	str8				DMA2 stream 0 transfer complete
			tcif1	str9				DMA2 stream 1 transfer complete
			tcif2	str10				DMA2 stream 2 transfer complete
			tcif3	str11				DMA2 stream 3 transfer complete
			tcif4	str12				DMA2 stream 4 transfer complete
			tcif5	str13				DMA2 stream 5 transfer complete
			tcif6	str14				DMA2 stream 6 transfer complete
			tcif7	str15				DMA2 stream 7 transfer complete
-	-	-	NC	str16				-
MPU	AXIM	SDMMC1	dataend_trg	str17				End of data
		SDMMC2	dataend_trg	str18				End of data
MCU	MLAHB	SDMMC3	dataend_trg	str19				End of data

Table 98. MDMA connections (continued)

Source				Destination				Comment
Domain	Bus	Peripheral	Signal	Signal	Peripheral	Bus	Domain	
MPU	AXIM	FMC	data_tx_rx	str20	MDMA	AXIM	MPU	NAND data transfer (Tx or Rx) channel
			error_position	str21				NAND ECC/BCH Error channel
MPU	AHB6	QUADSPI	ft_trg	str22				FIFO threshold
			tc_trg	str23				Transfer complete
-	-	-	NC	str24				-
-	-	-	NC	str25				-
-	-	-	NC	str26				-
-	-	-	NC	str27				-
-	-	-	NC	str28				-
MPU	AHB5	CRYP1	crypin_dmabreq	str29				4 word request from input
			crypout_dmabreq	str30				4 word ready from output
MPU	AHB5	HASH1	dmabreq	str31				16 word request
MPU	APB5	USART1	dma_req_r	str32				Receive data
			dma_req_t	str33				Transmit data
MPU	APB5	SPI6	dmareq_rx	str34				Receive data
			dmareq_tx	str35				Transmit data
MPU	APB5	I2C4	rxdmareq	str36				Receive data
			txdmareq	str37				Transmit data
MPU	APB5	I2C6	rxdmareq	str38				Receive data
			txdmareq	str39				Transmit data
-	-	-	NC	str40				-
-	-	-	NC	str41				-
-	-	-	NC	str42				-
-	-	-	NC	str43				-
-	-	-	NC	str44				-
-	-	-	NC	str45				-
-	-	-	NC	str46				-
-	-	-	NC	str47				-

17 MDMA controller (MDMA)

17.1 MDMA introduction

The master direct memory access (MDMA) is used in order to provide high-speed data transfer between memory and memory, or between peripherals and memory. Data can be quickly moved by the MDMA without any CPU action. This keeps the CPU resources free for other operations.

The MDMA controller provides a master AXI interface for main memory and peripheral register access (system access port).

The MDMA works in conjunction with the standard DMA controllers (DMA1 or DMA2). It offers up to 32 channels, each dedicated to manage memory access requests from one of the DMA stream memory buffer or other peripherals (with integrated FIFO).

17.2 MDMA main features

- AXI master bus architecture.
- 32 channels
- Up to 40 hardware trigger sources
- Each channel request can be selected among any of the request sources. This selection is software-configurable and allows several peripherals to initiate DMA requests. The trigger selection can be automatically changed at the end of one block transfer.
- All channels are identical and can be connected either to a standard DMA or a peripheral request (acknowledge by data read/write) system or to a peripheral request/acknowledge system
- Each channel also supports software trigger.
- One 256-level memory buffer, split in two 128-level first-in, first-out (FIFO), that is used to store temporary the data to be transferred (in burst or single transfer mode), for one or two consecutive buffers. The FIFO stores the data that are transferred during the current channel block transfer (up to the block transfer size). The second FIFO can be used for the next buffer to be transferred, either for the same channel or for the next channel transfer.
- The priorities between DMA channels are software-programmable (four levels consisting of very-high, high, medium, low) or hardware in case of equality (for example, channel 0 has priority over channel 1)
- Independent source and destination transfer width (byte, half-word, word, double-word): when the data widths of the source and destination are not equal, the MDMA can pack/unpack the necessary data to optimize the bandwidth.
- The size and address increment for both source and destination can be independently selected.

Note: Based on this separation, some more advanced packing/unpacking operations are available at software level. As an example, 2 x 16-bit data blocks can be interleaved together using two MDMA channels, in the destination memory, by simply programming the two channels with an increment step of 4 bytes and a data size of 16 bits + a start address shifted by two between the two channels.

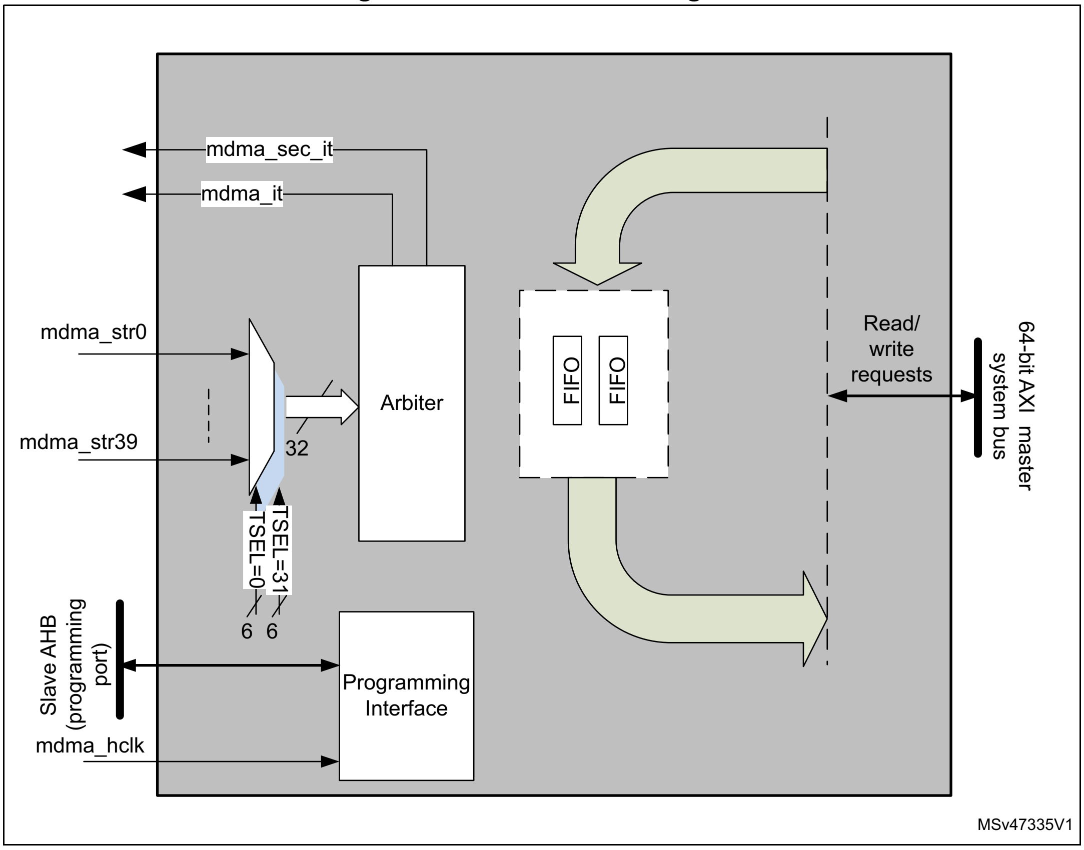
- Incrementing, decrementing or non incrementing/fixed addressing for source and destination
- Data packing/unpacking is always done respecting the little endian convention: lower address in a data entity (double-word, word, or half-word) contains always the lowest significant byte. This is independent of the address increment/decrement mode of both source and destination.
- Supports incremental burst transfers. The size of the burst is software-configurable, up to 128 bytes. For larger data sizes the burst length is limited, as to respect the maximum 128 bytes data burst size (such as 16 x 64 bits or 32 x 32 bits).
- For the TCM memory accesses, the burst access is only allowed when the increment and data size are identical and lower than or equal to 32 bits.
- Five event flags (MDMA channel transfer complete, MDMA block transfer complete, MDMA block repeat transfer Complete, MDMA buffer transfer Complete, MDMA transfer error) are available and can generate interrupts.

17.3 MDMA functional description

17.3.1 MDMA block diagram

The figure below shows the block diagram of the MDMA.

Figure 127. MDMA block diagram



17.3.2 MDMA internal signals

The table below shows the internal MDMA signals.

Table 99. MDMA internal input/output signals

Signal name	Signal type	Description
mdma_hclk	Digital input	MDMA AHB clock
mdma_it	Digital output	MDMA interrupt
mdma_sec_it	Digital output	MDMA secure interrupt
mdma_str[0:39]	Digital input	MDMA stream request

17.3.3 MDMA overview

The MDMA performs a direct memory transfer: as an AXI master, the MDMA can take the control of the AXI bus matrix to initiate AXI transactions.

The MDMA can carry out the following transactions:

- memory-to-memory (software triggered)
- peripheral-to-memory
- memory-to-peripheral

For the last two transaction types, the memory can also be replaced by a memory-mapped peripheral, which has no control over the MDMA flow. When these types of transaction are used and the request comes from a standard DMA (DMA1 or DMA2), the peripheral register access is replaced by a memory access to the memory buffer used by this DMA.

Note: Non-incrementing/decrementing mode is not used for memory accesses.

The source and destination are simply defined by the address (peripherals being memory-mapped also).

The AHB slave port is used to program the MDMA (it supports 8-,16-, and 32-bit accesses).

The size of the data array to be transferred for a single request is one of the following:

- buffer transfer size
- block size
- repeated block
- complete channel data (until the linked-list pointer for the channel is null)

The choice of the size is done through the TRGM[1:0] (Trigger mode) selection field.

The user must choose one of them based on the data array size available (usually in the DMA1/2 memory buffer) and the “real time” requirements for other MDMA channels (knowing that a buffer transfer is the minimum data aggregate to be transferred by the MDMA without doing a new arbitration between MDMA channel requests).

For each channel, there are three key data array sizes:

- Burst size: this is the length of the data transfer which can be performed in burst mode. This burst length defines the maximum transfer length which cannot be interrupted at bus arbitration level and can block other masters from accessing the bus.
- Buffer transfer size: this is the length of the data array to be transferred, on a channel, before checking for MDMA requests on other channels. This is the data array transfer lengths which cannot be interrupted at MDMA level (from other channel requests).
- Block size: this value has two meanings which can be used together:
  - main: this is length of the data block which is described in a block structure of the MDMA linked list (corresponds to one entry in the linked list)
  - selectable: when TRGM[1:0] equals 01, this is the length of the data array which is transferred on a single MDMA request activation (for the respective channel)

17.3.4 MDMA channel

Each MDMA channel provides an unidirectional transfer link between a source and a destination.

Each channel can perform transfer of the following types:

- Single block transfer: one block is transferred. At the end of the block, the MDMA channel is disabled and an end-of-channel transfer interrupt is generated.
- Repeated block transfer: a number of blocks is transferred before disabling the channel.
- Linked-list transfer: when the transfer of the current data block (or last block in a repeat) is completed, a new block control structure is loaded from memory and a new block transfer is started.

The minimum amount of data to be transferred for each request (buffer size, up to 128 bytes) is programmable. The total amount of data in a block, is programmable up to 64 Kbytes. This value is decremented after each transfer. When this counter reaches zero, the end of the block is reached and an action is taken based on the repeat counter (for repeated block transfer) and/or linked-list structure value.

Note: If the block length is not a multiple of the buffer length, the last buffer transfer in the block is shorter, covering the remaining bytes to be transferred in the current block.

If the link structure address points to a valid memory address, the MDMA reloads the whole channel descriptor structure register contents from memory at this address. Then, a new block transfer is then executed (on the next MDMA channel request) based on this information.

If the link structure address is 0x0, at the end of the current/repeated block transfer, the MDMA channel is disabled and the end-of-channel transfer interrupt is generated.

17.3.5 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4-Gbyte area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The source/destination addresses can be fixed (such as FIFO/single data register peripherals) or incremented/decremented. The transfer can be done in single access or in burst mode (programmable).

17.3.6 Pointer update

The source and destination memory pointers can optionally be automatically post-incremented/decremented, or kept constant after each transfer depending on the SINC[1:0] and DINC[1:0] bitfields in the MDMA_CxCR register.

Disabling the increment mode is useful when the peripheral source or destination data are accessed through a single register/FIFO mode.

If the increment/decrement mode is enabled, the address of the next data transfer is the address of the previous one incremented/decrement by 1, 2, 4 or 8 depending on the increment size programmed in SINCOS[1:0] or DINCOS[1:0] in MDMA_CxCR.

In order to optimize the packing operation, the increment offset size and the data size are programmable independently.

17.3.7 MDMA buffer transfer

This is the minimum logical amount of data (up to 128 bytes) which is transferred on an MDMA request event, on one channel.

An MDMA buffer transfer consists of a sequence of a given number of data transfers (done as single or burst data transfers). The number of data items to be transferred and their width (8, 16, 32, or 64 bits) are software programmable. The length of the burst used for data transfers is also programmable, independently.

After an event requiring a data array to be transferred, the DMA/peripheral sends a request signal to the MDMA, which serves the request depending on the channel priorities.

The request is acknowledged either by writing the mask data value to the address given mask address, when these registers are set or by hardware ACK signal.

If the mask address register is not set (0x00 value), the request can be reset by simply reading/writing the data to the peripheral. In this case, if the request is done by a destination peripheral, the write must be set as non bufferable, in order to avoid a false new MDMA request.

The total amount of data to be transferred, on the current channel, following a MDMA request, is determined by the TRGM[1:0] field.

If TRGM[1:0] equals 00, a single buffer is transferred, then the MDMA waits for another request on the same channel.

Note: In this case, the hardware request for the currently active channel (data in the FIFO) is not considered again until the end of the write phase for this channel. Even if the channel is still active at the end of the read phase, another channel (even with lower priority) can start the read phase. Lower priority channels can be interleaved with current channel transfer.

If TRGM[1:0] is different from 00 (multiple buffers need to be transferred), the mdma_strx for the current channel remains active (internally memorized) until the whole transfer defined by TRGM (block, repeated block or whole channel/linked list data) is completed.
After transferring an individual buffer, the MDMA enters in a new arbitration phase (between new external requests and internally memorized ones). If no other higher priority, the channel request is active, and a new buffer transfer is started for the same channel.

Note: When TRGM[1:0] is different from 00, a larger array of data is transferred for a single request. But, as the channel arbitration is done after each buffer transfer, no higher level MDMA requests are blocked for the more than a buffer transfer period, on any lower priority channel.

17.3.8 Request arbitration

An arbiter manages the MDMA channel requests based on their priority. When the MDMA is idle and after the end of each buffer transfer, all MDMA requests (hardware or software) are checked for all enabled channels.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the MDMA_CxCR register. There are four levels:
  - Very high priority
  - High priority
  - Medium priority
  - Low priority
- Hardware: at hardware level, the channel priority is fixed. If two requests have the same software priority level, the channel with the lower number takes priority over the stream with the higher number. For example, Channel 2 takes priority over Channel 4 when they have the same software priority level.

17.3.9 FIFO

A FIFO structure is used to temporarily store data coming from the source before writing them to the destination. There is a central FIFO structure which is used for all channels.

In order to maximize data bandwidth and bus usage, the following mechanisms are used, allowing multiple read/write operation to be executed in parallel.

- During a buffer transfer, as soon as the FIFO contains enough data for a destination burst transfer, the write operation starts.
- When the complete data for a buffer transfer has been read into the FIFO, the arbitration procedure is started. Following that, the next buffer data to be transferred can be read to the FIFO.

When an active channel is disabled due to an error, during a buffer transfer, the remaining data in the internal FIFO is discarded.

17.3.10 Block transfer

A block is a “contiguous” array of data, up to 64 Kbytes, which is transferred by successive buffer transfers.

Each block of data is defined by the start address and the block length. When a block transfer is completed, one of the following three actions can be executed:

- The block is part of a repeated block transfer: the block length is reloaded and new block start address is computed (based on the information in MDMA_CxBRUR).
- It is a single block or the last block in a repeated block transfer: the next block information is loaded from the memory (using the linked list address information, from MDMA_CxLAR).
- It is the last block which needs to be transferred for the current MDMA channel (MDMA_CxLAR = 0): the channel is disabled and no further MDMA requests are accepted for this channel.

17.3.11 Block repeat mode

The block repeat mode is used to repeat a block transfer, with different start addresses for source and destination.

When the repeat block mode is active (repeat counter ≠ 0), at the end of the current block transfer, the block parameters are updated (BNDT value reloaded and SAR/DAR values updated according to BRSUM/BRDUM configuration), and the repeat counter decremented by one.

When the repeat block counter reaches 0, this last block is treated as a single block transfer.

17.3.12 Linked-list mode

The linked-list mode allows a new MDMA configuration to be loaded (MDMA_CxTCR, MDMA_CxBNDTR, MDMA_CxSAR, MDMA_CxDAR, MDMA_CxBRUR, MDMA_CxLAR, MDMA_CxTBR, MDMA_CxMAR and MDMA_CxMDR registers), from the address given in MDMA_CxLAR. This address must address a memory mapped on the AXI system bus.

Following this operation, the channel is ready to accept new requests, as defined in the block/repeated block modes above, or continue the transfer if TRGM[1:0] = 11.

The trigger source can be automatically changed, when loading the MDMA_CxTBR value.

The TRGM and SWRM values must not be changed when TRGM[1:0] = 11.

17.3.13 MDMA transfer completion

Different events can generate an end of transfer by setting the CTCIF bit in the status register (MDMA_CxISR):

- The MDMA_CxBNDTR counter has reached zero, the Block Repeat Counter is 0 and the Link list pointer address is 0
- The channel is disabled before the end of transfer (by clearing the EN bit in MDMA_CxCR) and all the remaining data have been transferred from the FIFO to the destination

17.3.14 MDMA transfer suspension

At any time, a MDMA transfer can be suspended in order to be to be restarted later on or to be definitively disabled before the end of the MDMA transfer.

There are two cases:

- The channel is disabled, with no later-on restart from the point where it was stopped. There is no particular action to do, besides clearing the EN bit in the MDMA_CxCR register to disable the channel. The stream can take time to be disabled (on going buffer transfer is completed first). The transfer complete interrupt flag is set in order to indicate the end of transfer. The value of the EN bit in MDMA_CxCR is now 0 to confirm the channel interruption. The MDMA_CxNDTR register contains the number of remaining data items when the channel was stopped. The software can then determine how many data items have been transferred before the channel was interrupted.
- The channel is suspended before the number of remaining bytes to be transferred in the MDMA_CxBNDTR register reaches zero. The aim is to restart the transfer later by re-enabling the channel. The channel transfer complete interrupt flag CTCIF is set in order to indicate the end of transfer. If the MDMA_CxBNDTR, MDMA_CxSAR and

MDMA_CxDAR registers are not modified by software, the transfer continues when the channel is re-enabled. CTCIF must also be reset before restarting the channel.

Note: If the completed buffer is the last of the block, the configuration registers are also updated before disabling the channel, in order to be correctly prepared for a soft restart.

Before reprogramming the channel, the software must wait the MDMA_CTCIF register is set, in order to guarantee that any ongoing operation has been completed.

17.3.15 Error management

The MDMA can detect the following errors and the transfer error interrupt flag (TEIF) is set:

- when a bus error occurs during a MDMA read/write access
- when an address alignment does not correspond to the data size
- when the block size is not a multiple of the data size (for source and/or destination): this error is activated on the last transfer and the error address points to the last transfer (which cannot be done)

17.4 MDMA interrupts

For each MDMA channel, an interrupt can be produced on the following events:

- Channel transfer completed
- Block-transfer completed
- Block-transfer repeat completed
- Buffer transfer completed
- Transfer error

Separate interrupt enable control bits are available for flexibility as shown in the table below.

Table 100. MDMA interrupt requests

Interrupt event	Event flag	Enable control bit
Channel transfer completed	CTCIF	CTCIE
Block-transfer repeat completed	BTRIF	BTRIE
Block-transfer completed	BTIF	BTIE
Buffer transfer completed	TCIF	TCIE
Transfer error	TEIF	TEIE

Note: Before setting an enable control bit to 1, the corresponding event flag must be cleared, otherwise an interrupt may be immediately generated, if the bit is already set.

When at least one interrupt flag and the respective enable control bit are set, the channel interrupt bit is set in the MDMA_SGISR. The interrupt output or the secure interrupt output is also activated. This generates an interrupt if the respective interrupt channel is enabled in the NVIC.

17.5 MDMA registers

The MDMA registers can be accessed in word, half-word, or byte format.

17.5.1 MDMA global interrupt status register (MDMA_GISR0)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GIF31	GIF30	GIF29	GIF28	GIF27	GIF26	GIF25	GIF24	GIF23	GIF22	GIF21	GIF20	GIF19	GIF18	GIF17	GIF16
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GIF15	GIF14	GIF13	GIF12	GIF11	GIF10	GIF9	GIF8	GIF7	GIF6	GIF5	GIF4	GIF3	GIF2	GIF1	GIF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **GIF[31:0]**: Channel x global interrupt flag (x = 31 to 0)

This bitfield is set and reset by hardware. It is a logical OR of all channel x interrupt flags (CTCIF, BTIF, BRTIF, TEIF) which are enabled in MDMA_CTCIEx, MDMA_BTIEx, MDMA_BRTIEx, and MDMA_TEIEx.

0: No interrupt generated by channel x

1: Interrupt generated by channel x

17.5.2 MDMA secure global interrupt status register (MDMA_SGISR0)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GIF31	GIF30	GIF29	GIF28	GIF27	GIF26	GIF25	GIF24	GIF23	GIF22	GIF21	GIF20	GIF19	GIF18	GIF17	GIF16
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GIF15	GIF14	GIF13	GIF12	GIF11	GIF10	GIF9	GIF8	GIF7	GIF6	GIF5	GIF4	GIF3	GIF2	GIF1	GIF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **GIF[31:0]**: Channel x global interrupt flag (x = 31 to 0)

This bitfield is set and reset by hardware. It is a logical OR of all channel x interrupt flags (CTCIF, BTIF, BRTIF, TEIF) which are enabled in MDMA_CTCIEx, MDMA_BTIEx, MDMA_BRTIEx, and MDMA_TEIEx).

0: No interrupt generated by channel x

1: Interrupt generated by channel x

17.5.3 MDMA channel x interrupt status register (MDMA_CxISR)

Address offset: 0x40 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRQA
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCIF	BTIF	BRTIF	CTCIF	TEIF
											r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CRQA**: Channel x request active flag

This bit is set by software writing 1 to SWRQx in MDMA_CxCR, in order to request an MDMA transfer, and the channel x is enabled.

This bit is also set by hardware when the channel request becomes active, and the channel is enabled. The hardware request memorized until it is served.

This bit is cleared by hardware when the channel x request is completed (after the source write phase of the last buffer transfer due for the current request).

0: The MDMA transfer mdma_strx is inactive for channel x.

1: The MDMA transfer mdma_strx is active for channel x

Note: This bit is also reset by hardware when the channel is disabled (in case of transfer error, or when reaching the end of the channel data transfer - repeat block = 0 and linked list pointer null-, or by software programming the channel enable bit to 0 before that).

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **TCIF**: Channel x buffer transfer complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in MDMA_IFCRy.

0: No buffer transfer complete event on channel x

1: A buffer transfer complete event occurred on channel x

TC is set when a single buffer is transferred. It is activated on each channel transfer request. This can be used as a debug feature (without interrupt), indicating that (at least) an MDMA buffer transfer has been generated since the last flag reset.

Bit 3 **BTIF**: Channel x block transfer complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in MDMA_IFCRy.

0: No block transfer complete event on channel x

1: A block transfer complete event occurred on channel x.

Bit 2 **BRTIF**: Channel x block repeat transfer complete interrupt flag

This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in MDMA_IFCRy.

0: No block repeat transfer complete event on channel x

1: A block repeat transfer complete event occurred on channel x

Bit 1 **CTCIF**: Channel x channel transfer complete interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in MDMA_IFCRy.
0: No channel transfer complete event on channel x
1: A channel transfer complete event occurred on channel x
CTC is set when the last block was transferred and the channel has been automatically disabled.
CTC is also set when the channel is suspended, as a result of writing EN bit to 0.

Bit 0 **TEIF**: Channel x transfer error interrupt flag
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in MDMA_IFCRy.
0: No transfer error on stream x
1: A transfer error occurred on stream x

17.5.4 MDMA channel x interrupt flag clear register (MDMA_CxIFCR)

Address offset: 0x44 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLTCIF	CBTIF	CBRTIF	CCTCIF	CTEIF
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CLTCIF**: Clear buffer transfer complete Interrupt flag for channel x
Writing 1 into this bit clears TCIF in MDMA_ISRy.

Bit 3 **CBTIF**: Channel x clear block transfer complete interrupt flag
Writing 1 into this bit clears BTIF in MDMA_ISRy.

Bit 2 **CBRTIF**: Channel x clear block repeat transfer complete interrupt flag
Writing 1 into this bit clears BRTIF in MDMA_ISRy.

Bit 1 **CCTCIF**: Clear channel transfer complete interrupt flag for channel x
Writing 1 into this bit clears CTCIF in MDMA_ISRy.

Bit 0 **CTEIF**: Channel x clear transfer error interrupt flag
Writing 1 into this bit clears TEIF in MDMA_ISRy.

17.5.5 MDMA channel x error status register (MDMA_CxESR)

Address offset: 0x48 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BSE	ASE	TEMD	TELD	TED	TEA[6:0]						
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **BSE**: Block size error

This bit is set by hardware, when the block size is not an integer multiple of the data size either for source or destination. TED indicates whether the problem is on the source or destination. This bit is cleared by software writing 1 to CTEIF in MDMA_IFCRy.

0: No block size error

1: Programmed block size is not an integer multiple of the data size.

Bit 10 **ASE**: Address size error

This bit is set by hardware, when the programmed address is not aligned with the data size. TED indicates whether the problem is on the source or destination. It is cleared by software writing 1 to CTEIF in MDMA_IFCRy.

0: No address size error

1: Programmed address is not coherent with the data size.

Bit 9 **TEMD**: Transfer error mask data

This bit is set by hardware, in case of a transfer error while writing the mask data. It is cleared by software writing 1 to CTEIF in MDMA_IFCRy.

0: No mask write access error

1: The last transfer error on the channel was a related to a write of the mask data.

Bit 8 **TELD**: Transfer error link data

This bit is set by hardware, in case of a transfer error while reading the block link data structure. It is cleared by software writing 1 to CTEIF in MDMA_IFCRy.

0: No link data read access error

1: The last transfer error on the channel was a related to a read of the link data structure.

Bit 7 **TED**: Transfer error direction

This bit is set and cleared by hardware, in case of an MDMA data transfer error.

0: The last transfer error on the channel was a related to a read access.

1: The last transfer error on the channel was a related to a write access.

Bits 6:0 **TEA[6:0]**: Transfer error address

These bits are set and cleared by hardware in case of an MDMA data transfer error. They are used in conjunction with TED. This field indicates the 7 LSB bits of the address which generated a transfer/access error.

This field can be used by software to retrieve the failing address, by adding this value (truncated to the buffer transfer length size) to the current SAR/DAR value.

Note: The SAR/DAR current value does not reflect this last address due to the FIFO management system. The SAR/DAR are only updated at the end of a (buffer) transfer (of TLEN+1 bytes).

This field is not set in case of a link data error.

17.5.6 MDMA channel x control register (MDMA_CxCR)

This register is used to control the concerned channel.

Address offset: 0x4C + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWRQ
															w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	WEX	HEX	BEX	Res.	Res.	Res.	SM	PL[1:0]		TCIE	BTIE	BRTIE	CTCIE	TEIE	EN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SWRQ**: Software request

Writing 1 into this bit sets the CRQA in MDMA_ISRy, activating the request on channel x.

Note: Either the whole MDMA_CxCR register or the 8-/16-bit register at the address offset: 0x4E + 0x40 × channel number, can be used for SWRQ activation.

In case of software request, acknowledge is not generated (neither hardware signal, nor MDMA_CxMAR write access).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WEX**: Word endianness exchange

This bit is set and cleared by software.
0: Little endianness preserved for words
1: Word order exchanged in double word
When this bit is set, the word order in the destination double word is reversed: higher address word contains the data read from the lower address of the source.
If the destination is not a double word, do not care of this bit value.

Note: This bit is protected and can be written only if EN is 0.

Bit 13 **HEX**: Half-word endianness exchange

This bit is set and cleared by software.
0: Little endianness preserved for half words
1: half-word order exchanged in each word
When this bit is set, the half-word order in each destination word is reversed: higher address half-word contains the data read from the lower address of the source.
If destination length is shorter than word, do not care of this bit value.

Note: This bit is protected and can be written only if EN is 0.

Bit 12 **BEX**: Byte endianness exchange

This bit is set and cleared by software.
0: Little endianness preserved for bytes
1: byte order exchanged in each half-word
When this bit is set, the byte order in each destination half-word is reversed: higher address word contains the data read from the lower address of the source.
If destination is byte, do not care of this bit value.

Note: This bit is protected and can be written only if EN is 0.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SM**: Secure mode enable

This bit is set and cleared by software. It can only be written when the AHB slave port is in secure mode.

0: Secure mode disabled

1: Secure mode enable

When this bit is read as 0, the software can program the configuration bits related to the current channel independently of the secure mode of the AHB slave interface.

When this bit is set to 1, all registers of the current channel are write protected when the secure mode of the AHB slave port is not active. All MDMA master transactions are done in secure mode.

Note: This bit is protected and can be written only if EN is 0.

Bits 7:6 **PL[1:0]**: Priority level

These bits are set and cleared by software.

00: Low

01: Medium

10: High

11: Very high

Note: These bits are protected and can be written only if EN is 0.

Bit 5 **TCIE**: Buffer transfer complete interrupt enable

This bit is set and cleared by software.

0: TC interrupt disabled

1: TC interrupt enabled

Bit 4 **BTIE**: Block transfer interrupt enable

This bit is set and cleared by software.

0: BT complete interrupt disabled

1: BT complete interrupt enabled

Bit 3 **BRTIE**: Block repeat transfer interrupt enable

This bit is set and cleared by software.

0: BT interrupt disabled

1: BT interrupt enabled

Bit 2 **CTCIE**: Channel transfer complete interrupt enable

This bit is set and cleared by software.

0: TC interrupt disabled

1: TC interrupt enabled

Bit 1 **TEIE**: Transfer error interrupt enable

This bit is set and cleared by software.

0: TE interrupt disabled

1: TE interrupt enabled

Bit 0 **EN**: Channel enable/flag channel ready when read low

This bit is set and cleared by software.

0: Channel disabled

1: Channel enabled

This bit can be cleared by hardware:

- on a MDMA end of transfer (stream ready to be configured)
- if a transfer error occurs on the AXI master buses (bus error/HardFault)
- if another error condition is encountered (data alignment, block/data size incompatibility)

When this bit is reset by software, the ongoing buffer transfer (if any) is completed.
All status/configuration registers keep their current values. If the channel is re-enabled without writing these registers, the channel continues from the point where it was interrupted.

When this bit is read as 0, the software is allowed to program the configuration registers.
It is forbidden to write these registers when the EN bit is read as 1 (writes are ignored).

Note: When this bit is reset by software, it is recommended to wait for the CTCIF = 1, in order to ensure that any ongoing buffer transfer has been completed, before reprogramming the channel.

17.5.7 MDMA channel x transfer configuration register (MDMA_CxTCR)

Address offset: 0x50 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

This register is used to configure the concerned channel. In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from the memory (from address given by current LAR[31:0] + 0x00).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWM	SWRM	TRGM[1:0]		PAM[1:0]		PKE	TLEN[6:0]							DBURST[2:1]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBURST[0]	SBURST[2:0]			DINCOS[1:0]		SINCOS[1:0]		DSIZE[1:0]		SSIZE[1:0]		DINC[1:0]		SINC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BWM**: Bufferable write mode

This bit is set and cleared by software.

0: The destination write operation is non-bufferable.

1: The destination write operation is bufferable.

Note: This bit is protected and can be written only if EN is 0.

All MDMA destination accesses are non-cacheable.

Bit 30 **SWRM**: Software request mode

This bit is set and cleared by software. If an hardware or software request is currently active, the bit change is delayed until the current transfer is completed.

0: Hardware request are taken into account: the transfer is initiated as defined by TRGM value and acknowledged by the MDMA ACKx signal. If the MDMA_CxMAR contains a valid address, the MDMA_CxMDR value is also written at MDMA_CxMAR address.

1: Hardware request are ignored. Transfer is triggered by software writing 1 to the SWRQ bit.

Note: This bit is protected and can be written only if EN is 0.

Bits 29:28 **TRGM[1:0]**: Trigger mode

These bits are set and cleared by software.

00: Each MDMA request (software or hardware) triggers a buffer transfer.

01: Each MDMA request (software or hardware) triggers a block transfer.

10: Each MDMA request (software or hardware) triggers a repeated block transfer (if the block repeat is 0, a single block is transferred).

11: Each MDMA request (software or hardware) triggers the transfer of the whole data for the respective channel (for example linked list) until the channel reach the end and it is disabled.

Note: If TRGM is 11 for the current block, all values loaded at the end of the current block through the linked-list mechanism must keep the same value (TRGM = 11), and the same SWRM value, otherwise the result is undefined.

These bits are protected and can be written only if EN is 0.

Bits 27:26 **PAM[1:0]**: Padding/alignment mode

These bits are set and cleared by software.

00: Right aligned, padded with 0s (default). If source data is larger than destination size, only LSB part of the source is written to the destination address. The reminder part is discarded.

01: Right aligned, sign extended

10: Left aligned (padded with 0s). if source data is larger than destination size, only MSB part of the source is written to the destination address. The reminder part is discarded.

11: Reserved

Note: When PKE = 1 or DSIZE=SSIZE, these bits are ignored.

These bits are protected and can be written only if EN is 0.

Bit 25 **PKE**: Pack enable

This bit is set and cleared by software.

0: The source data is written to the destination as is. If source size is smaller than destination, it is padded according to the PAM value. If source data size is larger than destination one, it is truncated. The alignment is done according to the PAM[1:0] value.

1: The source data is packed/unpacked into the destination data size. All data are right aligned, in little-endian mode.

Note: This bit is protected and can be written only if EN is 0.

Bits 24:18 **TLEN[6:0]**: Buffer transfer length (number of bytes - 1)

These bits are set and cleared by software. TLEN + 1 value represents the number of bytes to be transferred in a single transfer.

The transfer length must be a multiple of the data size (for both source and destination)

When the source/destination sizes are different and padding/truncation is used, TLEN + 1 refers to the source data array size.

Note: These bits are protected and can be written only if EN is 0.

DBURST value must be programmed in order to ensure that the burst size is lower than the transfer size.

Bits 17:15 **DBURST[2:0]**: Destination burst transfer configuration

These bits are set and cleared by software.

000: Single transfer

N: burst of 2^N beats

DBURST value must be programmed as to ensure that the burst size is lower than the transfer length. If this is not ensured, the result is unpredictable.

Note: These bits are protected and can be written only if EN is 0.

When DINC = 00, DBURST must be maximum 100 (burst of 16), else the result is unpredictable.

Bits 14:12 **SBURST[2:0]**: Source burst transfer configuration

These bits are set and cleared by software.

000: single transfer

N: burst of 2^N beats

SBURST value must be programmed as to ensure that the burst size is lower than the transfer length. If this is not ensured, the result is unpredictable.

Note: These bits are protected and can be written only if EN is 0.

When SINC = 00, SBURST must be maximum 100 (burst of 16), else the result is unpredictable.

Bits 11:10 **DINCOS[1:0]**: Destination increment offset size

These bits are set and cleared by software.

00: byte (8-bit)

01: half-word (16-bit)

10: word (32-bit)

11: Double-Word (64-bit) -

This bits have no meaning if bit DINC[1:0] = 00.

Note: If DINCOS < DSIZE and DINC ≠ 00, the result is unpredictable.

These bits are protected and can be written only if EN = 0.

Bits 9:8 **SINCOS[1:0]**: Source increment offset size

These bits are set and cleared by software.

00: byte (8-bit)

01: half-word (16-bit)

10: word (32-bit)

11: Double-Word (64-bit) -

This bits have no meaning if bit SINC[1:0] = 00.

Note: If SINCOS < SSIZE and SINC ≠ 00, the result is unpredictable.

These bits are protected and can be written only if EN = 0.

Bits 7:6 **DSIZE[1:0]**: Destination data size

These bits are set and cleared by software.

00: Byte (8 bits)

01: Half-word (16 bits)

10: Word (32 bits)

11: Double-word (64 bits)

Note: If DINCOS < DSIZE and DINC ≠ 00, the result is unpredictable.

These bits are protected and can be written only if EN = 0.

Bits 5:4 **SSIZE[1:0]**: Source data size

These bits are set and cleared by software.

00: Byte (8 bits)

01: Half-word (16 bits)

10: Word (32 bits)

11: Double-word (64 bits)

Note: If SINCOS < SSIZE and SINC ≠ 00, the result is unpredictable.

These bits are protected and can be written only if EN is 0.

Bits 3:2 **DINC[1:0]**: Destination increment mode
These bits are set and cleared by software.
00: Destination address pointer is fixed.
10: Destination address pointer is incremented after each data transfer (increment is done according to DINCOS).
11: Destination address pointer is decremented after each data transfer (increment is done according to DINCOS).
These bits are protected and can be written only if EN = 0.

Bits 1:0 **SINC[1:0]**: Source increment mode
These bits are set and cleared by software.
00: Source address pointer is fixed.
10: Source address pointer is incremented after each data transfer (increment is done according to SINCOS).
11: Source address pointer is decremented after each data transfer (decrement is done according to SINCOS).
These bits are protected and can be written only if EN = 0.

17.5.8 MDMA channel x block number of data register (MDMA_CxBNDTR)

Address offset: 0x54 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from the memory (from address given by current LAR[31:0] + 0x04).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BRC[11:0]												BRDUM	BRSUM	Res.	BNDT[16]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 **BRC[11:0]**: Block repeat count
This field contains the number of repetitions of the current block (0 to 4095). When the channel is enabled, this register is read-only, indicating the remaining number of blocks, excluding the current one. This register decrements after each complete block transfer.
Once the last block transfer has completed, this register can either stay at zero or be reloaded automatically from memory (in linked-list mode, meaning link address valid).
Note: These bits are protected and can be written only if EN = 0.

Bit 19 **BRDUM**: Block repeat destination address update mode
0: At the end of a block transfer, the MDMA_DAR register is updated by adding the DUV to the current DAR value (current destination address).
1: At the end of a block transfer, the MDMA_DAR register is updated by subtracting the DUV from the current DAR value (current destination address).
Note: These bits are protected and can be written only if EN = 0.

Bit 18 **BRSUM**: Block repeat source address update mode

0: At the end of a block transfer, the MDMA_SAR register is updated by adding the SUV to the current SAR value (current source address).

1: At the end of a block transfer, the MDMA_SAR register is updated by subtracting the SUV from the current SAR value (current source address).

Note: These bits are protected and can be written only if EN = 0.

Bit 17 Reserved, must be kept at reset value.

Bits 16:0 **BNDT[16:0]**: Block number of data bytes to transfer

Number of bytes to be transferred (0 up to 65536) in the current block. When the channel is enabled, this register is read-only, indicating the remaining data items to be transmitted. During the channel activity, this register decrements, indicating the number of data items remaining in the current block.

Once the block transfer has completed, this register can either stay at zero or be reloaded automatically with the previously programmed value if the channel is configured in block repeat mode.

If the value of this register is zero, no transaction can be served even if the stream is enabled.

Note: If the BNDT value is not an integer multiple of the TLEN + 1 value, the last transfer is shorter and contains only the remaining data in the block.

The block size must be a multiple of the source and destination data size. If this is not true, an error is set and the no data are written

These bits are protected and can be written only if EN = 0.

17.5.9 MDMA channel x source address register (MDMA_CxSAR)

Address offset: 0x58 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x08).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SAR[31:0]**: Source address

These bits represent the base address of the peripheral data register from/to which the data is read. They must be aligned with the SSIZE (SAR[1:0] = 00 when SSIZE = 10), but may be unaligned with the SINCOS.

During the channel activity, this register is updated, reflecting the current address from which the data is read next.

When the block repeat mode is active, when a block transfer is completed, the source address is updated by adding/subtracting the SAU value to the current value (already updated after the last transfer in the block).

When the linked-list mode is active, at the end of a block (repeated or not) transfer, the SAR value is loaded from memory (from address LSA + m).

Note: These bits are write-protected and can be written only when bit EN = 0.

17.5.10 MDMA channel x destination address register (MDMA_CxDAR)

Address offset: 0x5C + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x0C).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DAR[31:0]**: Destination address

Base address of the destination address to which the data is written. Must be aligned with the DSIZE (for example DAR[0] = 0 when DSIZE=01), but may be unaligned with the DINCOS.

During the channel activity, this register is updated, reflecting the current address to which the data is written next.

When the block repeat mode is active, when a block transfer is completed, the Destination address is updated by adding/subtracting the DAU value to the current value (after the last transfer in the block).

When the linked-list mode is active, at the end of a block (repeated or not) transfer, the DAR value is loaded from memory (from address LSA + m).

Note: These bits are write-protected and can be written only when bit EN = 0.

17.5.11 MDMA channel x block repeat address update register (MDMA_CxBRUR)

Address offset: 0x60 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x10).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DUV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **DUV[15:0]**: Destination address update value

This value is used to update (by addition or subtraction) the current destination address at the end of a block transfer. It must be an integer multiple of DSIZE, in order to keep DAR aligned to DSIZE (DAR[1:0] = 00 when DSIZ E =10).

If this value is 0, the next repetition of the block transfer continues to the next address.

When the block repeat mode is not active (BRC = 0), this field is ignored.

Note: These bits are write-protected and can be written only when bit EN = 0.

This field must be programmed to 0 when DINC[1:0] = 00.

Bits 15:0 **SUV[15:0]**: Source address update value

This value is used to update (by addition or subtraction) the current source address at the end of a block transfer. It must be an integer multiple of SSIZE, in order to keep SAR aligned to SSIZE (SAR[1:0] = 00 when SSIZE = 10).

If this value is 0, the next repetition of the block transfer continues from the next address.

When the block repeat mode is not active (BRC=0), this field is ignored.

Note: These bits are write-protected and can be written only when bit EN = 0.

This field must be programmed to 0 when SINC[1:0] = 00.

17.5.12 MDMA channel x link address register (MDMA_CxLAR)

Address offset: 0x64 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x14).

Note: The new value is only taken into account after all registers are updated, for the next end of block.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **LAR[31:0]**: Link address register

At the end of a (repeated) block transfer, the current channel configuration registers (MDMA_CxTCR, MDMA_CxBNDTR, MDMA_CxSAR, MDMA_CxDAR, MDMA_CxBRUR, MDMA_CxMAR, MDMA_CxMDR and MDMA_CxLAR itself) are loaded with the data structure found at this address.

If the value of this register is 0, no register update takes place, the channel is disabled and CTCIF is set, indicating the end of the transfer for this channel.

The channel configuration (LAR address) must be in the AXI address space.

LAR value must be aligned at a double-word address (LAR[2:0] = 0x0).

Note: These bits are write-protected and can be written only when bit EN = 0.

17.5.13 MDMA channel x trigger and bus selection register (MDMA_CxTBR)

Address offset: 0x68 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x18).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSEL[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TSEL[5:0]**: Trigger selection

This field selects the hardware trigger (RQ) input for channel x. The ACK is sent on the ACK output having the same index value.

When SWRM bit is set (software request selected), this field is ignored.

Note: These bits are write-protected and can be written only when bit EN = 0.

If multiple channels are triggered by the same event (have the same TSEL value), all of them are triggered in parallel. Only the channel with the lowest index acknowledges the request (as software or hardware acknowledge).

17.5.14 MDMA channel x mask address register (MDMA_CxMAR)

Address offset: 0x70 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In linked-list mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x20).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MAR[31:0]**: Mask address

When an ACK signal is generated, a write of the MDR value is also done to this address. This is used to clear the RQ signal generated by the DMA2 by writing to its interrupt clear register.

If the value of this register is 0, this function is disabled.

Note: These bits are write-protected and can be written only when bit EN = 0.

17.5.15 MDMA channel x mask data register (MDMA_CxMDR)

Address offset: 0x74 + 0x40 * x, (x = 0 to 31)

Reset value: 0x0000 0000

In Linked List mode, at the end of a block (single or last block in repeated block transfer mode), this register is loaded from memory (from address given by current LAR[31:0] + 0x24).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MDR[31:0]**: Mask data

When an ACK signal is generated, a write of the MDR value is also done to the address defined by the MDMA_MAR register. This is used to clear the RQ signal generated by the DMA2 by writing to its interrupt clear register.

Note: These bits are write-protected and can be written only when bit EN = 0.

17.5.16 MDMA register map

Table 101. MDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	MDMA_GISR0	GIF31	GIF30	GIF29	GIF28	GIF27	GIF26	GIF25	GIF24	GIF23	GIF22	GIF21	GIF20	GIF19	GIF18	GIF17	GIF16	GIF15	GIF14	GIF13	GIF12	GIF11	GIF10	GIF9	GIF8	GIF7	GIF6	GIF5	GIF4	GIF3	GIF2	GIF1	GIF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Reserved																															
0x08	MDMA_SGISR0	GIF31	GIF30	GIF29	GIF28	GIF27	GIF26	GIF25	GIF24	GIF23	GIF22	GIF21	GIF20	GIF19	GIF18	GIF17	GIF16	GIF15	GIF14	GIF13	GIF12	GIF11	GIF10	GIF9	GIF8	GIF7	GIF6	GIF5	GIF4	GIF3	GIF2	GIF1	GIF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C - 0x3C	Reserved	Reserved																															
0x40 + 0x40 * x (x=0 to 31)	MDMA_CxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCIF	BTIF	BRTIF	CTCIF	TEIF
	Reset value																												0	0	0	0	0
0x44 + 0x40 * x (x=0 to 31)	MDMA_CxIFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLTCIF	CBTIF	CBRTIF	CCTCIF	CTEIF
	Reset value																												0	0	0	0	0
0x48 + 0x40 * x (x=0 to 31)	MDMA_CxESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSE	ASE	TEMD	TELD	TED	TEA[6:0]						
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x4C + 0x40 * x (x=0 to 31)	MDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWRQ	Res.	WEX	HEX	BEX	Res.	Res.	Res.	SM	PL[1:0]		TCIE	BTIE	BRTIE	CTCIE	TEIE	EN
	Reset value																0		0	0	0				0	0	0	0	0	0	0	0	0
0x50 + 0x40 * x (x=0 to 31)	MDMA_CxTCR	BWM	SWRM	TRGM [1:0]		PAM[1:0]		PKE	TLEN[6:0].							DBURST [2:0]			SBURST [2:0]			DINCOS [1:0]		SINCOS [1:0]		DSIZE [1:0]		SSIZE [1:0]		DINC [1:0]		SINC [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 101. MDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54 + 0x40 * x (x=0 to 31)	MDMA_CxBNDTR	BRC[11:0]												BRDUM	BRSUM	Res.	BNDT[16:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58 + 0x40 * x (x=0 to 31)	MDMA_CxSAR	SAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C + 0x40 * x (x=0 to 31)	MDMA_CxDAR	DAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60 + 0x40 * x (x=0 to 31)	MDMA_CxBRUR	DUV[15:0].																SUV[15:0].															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64 + 0x40 * x (x=0 to 31)	MDMA_CxLAR	LAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68 + 0x40 * x (x=0 to 31)	MDMA_CxTBR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res.	TSEL[5:0]					
	Reset value																											0	0	0	0	0	0
0x6C+ 0x40 * x (x=0 to 31)	Reserved	Reserved																															
0x70 + 0x40 * x (x=0 to 31)	MDMA_CxMAR	MAR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 + 0x40 * x (x=0 to 31)	MDMA_CxMDR	MDR[31:0].																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5* for the register boundary addresses.

18 Direct memory access controller (DMA)

18.1 DMA introduction

Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. Data can be quickly moved by DMA without any CPU action. This keeps CPU resources free for other operations.

The DMA controller combines a powerful dual AHB master bus architecture with independent FIFO to optimize the bandwidth of the system, based on a complex bus matrix architecture.

The two DMA controllers (DMA1, DMA2) have 8 streams each, dedicated to managing memory access requests from one or more peripherals.

Each DMA stream is driven by one DMAMUX output channel (request). Any DMAMUX output request can be individually programmed in order to select the DMA request source signal, from any of the 116 available request input signals.

Refer to the *Section 19.3: DMAMUX implementation* for more information about the DMA requests and streams mapping.

Each DMA controller has an arbiter for handling the priority between DMA requests.

18.2 DMA main features

The main DMA features are:

- Dual AHB master bus architecture, one dedicated to memory accesses and one dedicated to peripheral accesses
- AHB slave programming interface supporting only 32-bit accesses
- 8 streams for each DMA controller, up to 116 channels (requests) per stream
- Four-word depth 32 first-in, first-out memory buffers (FIFOs) per stream, that can be used in FIFO mode or direct mode:
  - FIFO mode: with threshold level software selectable between 1/4, 1/2 or 3/4 of the FIFO size
  - Direct mode: each DMA request immediately initiates a transfer from/to the memory. When it is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads only one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.
- Each stream can be configured to be:
  - a regular channel that supports peripheral-to-memory, memory-to-peripheral and memory-to-memory transfers
  - a double buffer channel that also supports double buffering on the memory side
- Priorities between DMA stream requests are software-programmable (four levels consisting of very high, high, medium, low) or hardware in case of equality (for example, request 0 has priority over request 1)
- Each stream also supports software trigger for memory-to-memory transfers

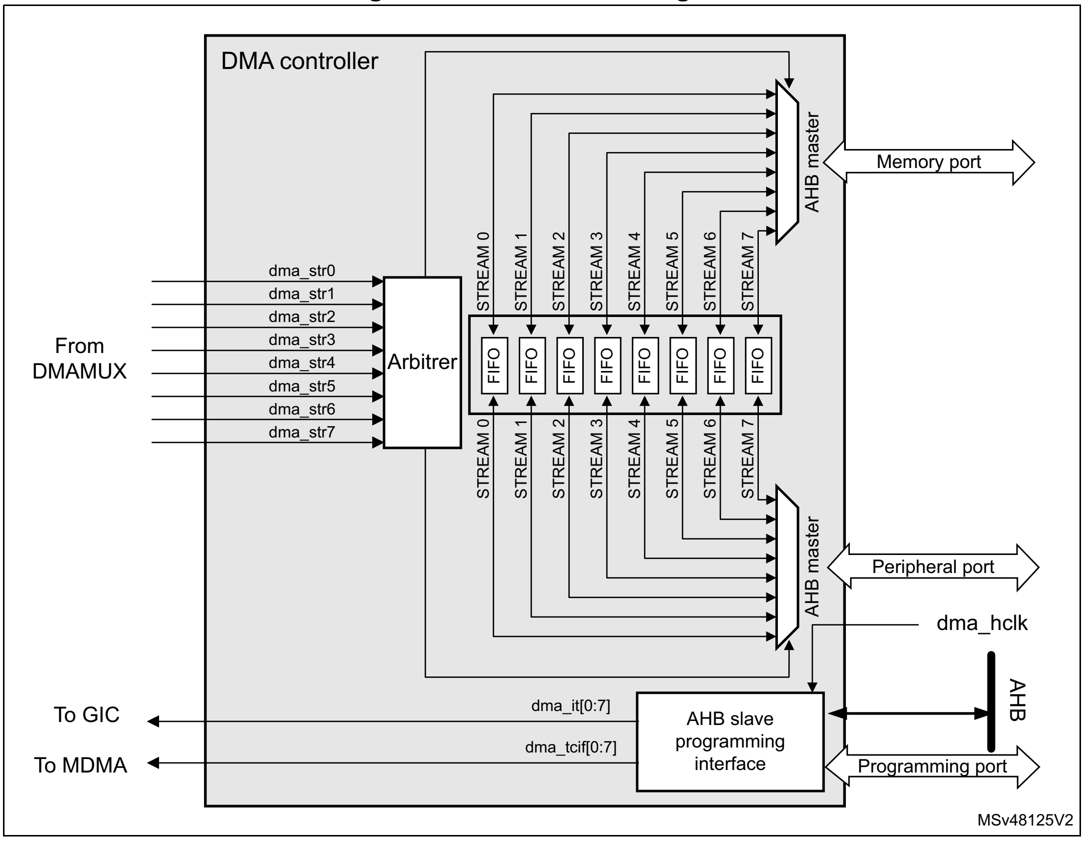
- Each stream request can be selected among up to 116 possible channel requests. This selection is software-configurable by the DMAMUX and allows 108 peripherals to initiate DMA requests
- The number of data items to be transferred can be managed either by the DMA controller or by the peripheral:
  - DMA flow controller: the number of data items to be transferred is software-programmable from 1 to 65535
  - Peripheral flow controller: the number of data items to be transferred is unknown and controlled by the source or the destination peripheral that signals the end of the transfer by hardware
- Independent source and destination transfer width (byte, half-word, word): when the data widths of the source and destination are not equal, the DMA automatically packs/unpacks the necessary transfers to optimize the bandwidth. This feature is only available in FIFO mode
- Incrementing or non-incrementing addressing for source and destination
- Supports incremental burst transfers of 4, 8 or 16 beats. The size of the burst is software-configurable, usually equal to half the FIFO size of the peripheral
- Each stream supports circular buffer management
- 5 event flags (DMA half transfer, DMA transfer complete, DMA transfer error, DMA FIFO error, direct mode error) logically ORed together in a single interrupt request for each stream

18.3 DMA functional description

18.3.1 DMA block diagram

The figure below shows the block diagram of a DMA.

Figure 128. DMA block diagram



18.3.2 DMA overview

The DMA controller performs direct memory transfer: as an AHB master, the DMA controller can take the control of the AHB bus matrix to initiate AHB transactions.

The DMA controller carries out the following transactions:

- peripheral-to-memory
- memory-to-peripheral
- memory-to-memory

The DMA controller provides two AHB master ports: the AHB memory port, intended to be connected to memories and the AHB peripheral port, intended to be connected to peripherals. However, to allow memory-to-memory transfers, the AHB peripheral port must also have access to the memories.

The AHB slave port is used to program the DMA controller (it supports only 32-bit accesses).

18.3.3 DMA transactions

A DMA transaction consists of a sequence of a given number of data transfers. The number of data items to be transferred and their width (8-bit, 16-bit or 32-bit) are software-programmable.

Each DMA transfer consists of three operations:

- a loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register
- a storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register
- a post-decrement of the DMA_SxNDTR register, containing the number of transactions that still have to be performed

After an event, the peripheral sends a request signal to the DMA controller. The DMA controller serves the request depending on the channel priorities. As soon as the DMA controller accesses the peripheral, an Acknowledge signal is sent to the peripheral by the DMA controller. The peripheral releases its request as soon as it gets the Acknowledge signal from the DMA controller. Once the request has been deasserted by the peripheral, the DMA controller releases the Acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

18.3.4 DMA request mapping

The DMA request mapping from peripherals to DMA streams is described in *Section 19.3: DMAMUX implementation*.

18.3.5 Arbiter

An arbiter manages the 8 DMA stream requests based on their priority for each of the two AHB master ports (memory and peripheral ports) and launches the peripheral/memory access sequences.

Priorities are managed in two stages:

- Software: each stream priority can be configured in the DMA_SxCR register. There are four levels:
  – Very high priority
  – High priority
  – Medium priority
  – Low priority
- Hardware: If two requests have the same software priority level, the stream with the lower number takes priority over the stream with the higher number. For example, stream 2 takes priority over stream 4.

18.3.6 DMA streams

Each of the eight DMA controller streams provides a unidirectional transfer link between a source and a destination.

Each stream can be configured to perform:

- Regular type transactions: memory-to-peripherals, peripherals-to-memory or memory-to-memory transfers
- Double-buffer type transactions: double buffer transfers using two memory pointers for the memory (while the DMA is reading/writing from/to a buffer, the application can write/read to/from the other buffer).

The amount of data to be transferred (up to 65535) is programmable and related to the source width of the peripheral that requests the DMA transfer connected to the peripheral AHB port. The register that contains the amount of data items to be transferred is decremented after each transaction.

18.3.7 Source, destination and transfer modes

Both source and destination transfers can address peripherals and memories in the entire 4-Gbyte area, at addresses comprised between 0x0000 0000 and 0xFFFF FFFF.

The direction is configured using the DIR[1:0] bits in the DMA_SxCR register and offers three possibilities: memory-to-peripheral, peripheral-to-memory or memory-to-memory transfers.

The table below describes the corresponding source and destination addresses.

Table 102. Source and destination address

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR
01	Memory-to-peripheral	DMA_SxM0AR	DMA_SxPAR
10	Memory-to-memory	DMA_SxPAR	DMA_SxM0AR
11	Reserved	-	-

When the data width (programmed in the PSIZE or MSIZE bits in the DMA_SxCR register) is a half-word or a word, respectively, the peripheral or memory address written into the DMA_SxPAR or DMA_SxM0AR/M1AR registers has to be aligned on a word or half-word address boundary, respectively.

Peripheral-to-memory mode

Figure 129 describes this mode.

When this mode is enabled (by setting the bit EN in the DMA_SxCR register), each time a peripheral request occurs, the stream initiates a transfer from the source to fill the FIFO.

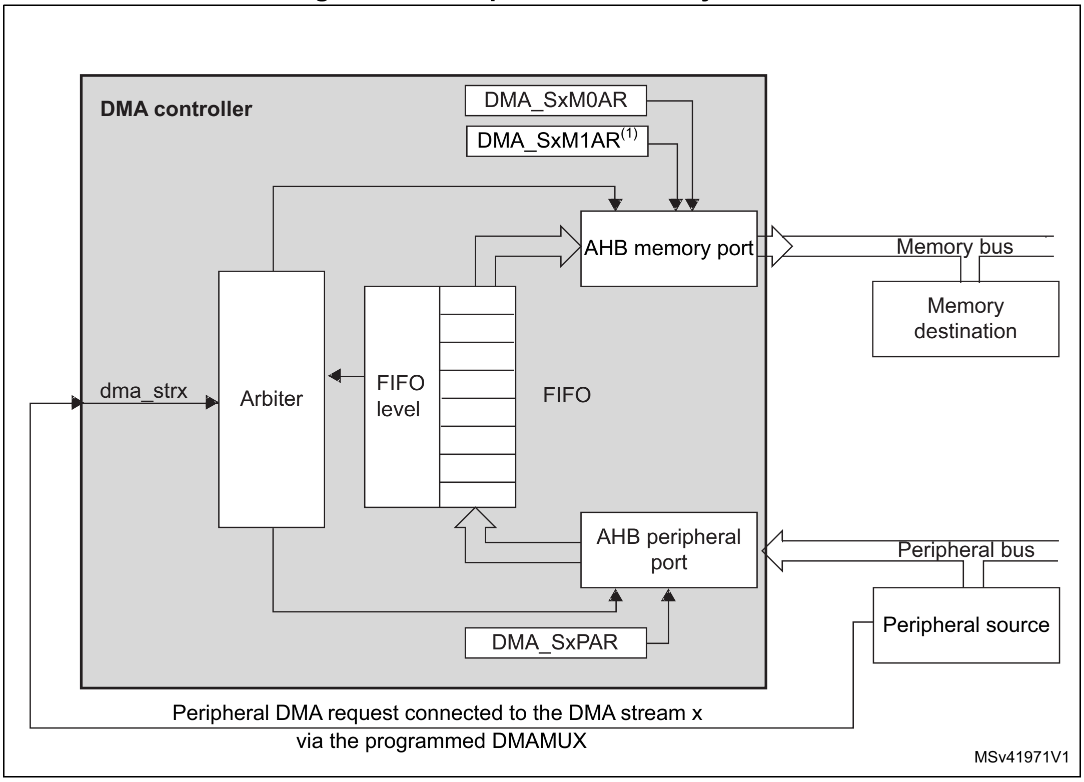
When the threshold level of the FIFO is reached, the contents of the FIFO are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is 0), the threshold level of the FIFO is not used: after each single data transfer from the peripheral to the FIFO, the corresponding data are immediately drained and stored into the destination.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 129. Peripheral-to-memory mode



1. For double-buffer mode.

Memory-to-peripheral mode

Figure 130 describes this mode.

When this mode is enabled (by setting the EN bit in the DMA_SxCR register), the stream immediately initiates transfers from the source to entirely fill the FIFO.

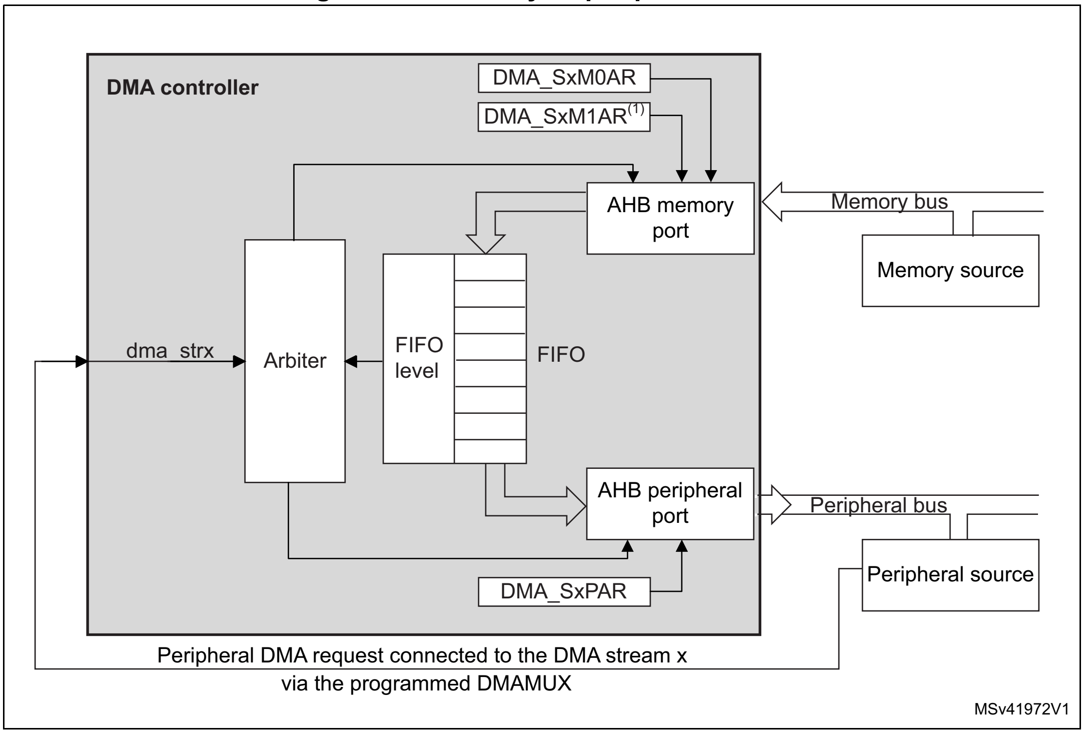
Each time a peripheral request occurs, the contents of the FIFO are drained and stored into the destination. When the level of the FIFO is lower than or equal to the predefined threshold level, the FIFO is fully reloaded with data from the memory.

The transfer stops once the DMA_SxNDTR register reaches zero, when the peripheral requests the end of transfers (in case of a peripheral flow controller) or when the EN bit in the DMA_SxCR register is cleared by software.

In direct mode (when the DMDIS value in the DMA_SxFCR register is 0), the threshold level of the FIFO is not used. Once the stream is enabled, the DMA preloads the first data to transfer into an internal FIFO. As soon as the peripheral requests a data transfer, the DMA transfers the preloaded value into the configured destination. It then reloads again the empty internal FIFO with the next data to be transfer. The preloaded data size corresponds to the value of the PSIZE bitfield in the DMA_SxCR register.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Figure 130. Memory-to-peripheral mode



1. For double-buffer mode.

Memory-to-memory mode

The DMA channels can also work without being triggered by a request from a peripheral. This is the memory-to-memory mode, described in *Figure 131*.

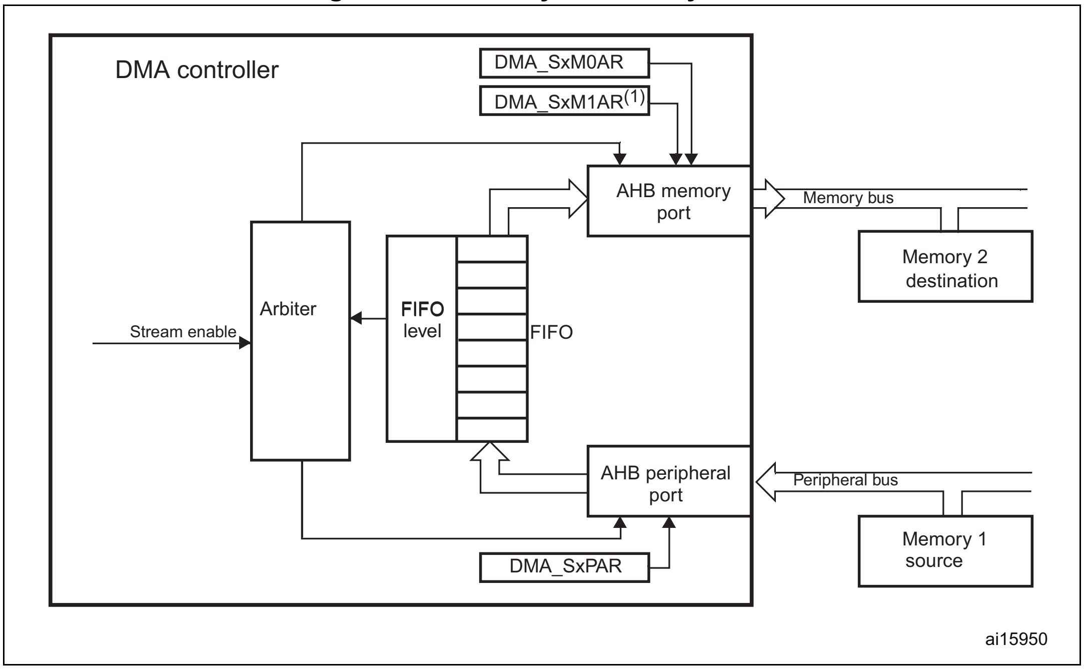
When the stream is enabled by setting the Enable bit (EN) in the DMA_SxCR register, the stream immediately starts to fill the FIFO up to the threshold level. When the threshold level is reached, the FIFO contents are drained and stored into the destination.

The transfer stops once the DMA_SxNDTR register reaches zero or when the EN bit in the DMA_SxCR register is cleared by software.

The stream has access to the AHB source or destination port only if the arbitration of the corresponding stream is won. This arbitration is performed using the priority defined for each stream using the PL[1:0] bits in the DMA_SxCR register.

Note: When memory-to-memory mode is used, the circular and direct modes are not allowed.

Figure 131. Memory-to-memory mode



1. For double-buffer mode.

18.3.8 Pointer incrementation

Peripheral and memory pointers can optionally be automatically post-incremented or kept constant after each transfer depending on the PINC and MINC bits in the DMA_SxCR register.

Disabling the increment mode is useful when the peripheral source or destination data is accessed through a single register.

If the increment mode is enabled, the address of the next transfer is the address of the previous one incremented by 1 (for bytes), 2 (for half-words) or 4 (for words) depending on the data width programmed in the PSIZE or MSIZE bits in the DMA_SxCR register.

In order to optimize the packing operation, it is possible to fix the increment offset size for the peripheral address whatever the size of the data transferred on the AHB peripheral port. The PINCOS bit in the DMA_SxCR register is used to align the increment offset size with the data size on the peripheral AHB port, or on a 32-bit address (the address is then incremented by 4). The PINCOS bit has an impact on the AHB peripheral port only.

If the PINCOS bit is set, the address of the following transfer is the address of the previous one incremented by 4 (automatically aligned on a 32-bit address), whatever the PSIZE value. The AHB memory port, however, is not impacted by this operation.

18.3.9 Circular mode

The circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the CIRC bit in the DMA_SxCR register.

When the circular mode is activated, the number of data items to be transferred is automatically reloaded with the initial value programmed during the stream configuration phase, and the DMA requests continue to be served.

Note: In the circular mode, it is mandatory to respect the following rule in case of a burst mode configured for memory:

DMA_SxNDTR = Multiple of ((Mburst beat) × (Msize)/(Psize)), where:

- *(Mburst beat) = 4, 8 or 16 (depending on the MBURST bits in the DMA_SxCR register)*
- *((Msize)/(Psize)) = 1, 2, 4, 1/2 or 1/4 (Msize and Psize represent the MSIZE and PSIZE bits in the DMA_SxCR register. They are byte dependent)*
- *DMA_SxNDTR = Number of data items to transfer on the AHB peripheral port*

For example: Mburst beat = 8 (INCR8), MSIZE = 00 (byte) and PSIZE = 01 (half-word), in this case: DMA_SxNDTR must be a multiple of (8 × 1/2 = 4).

If this formula is not respected, the DMA behavior and data integrity are not guaranteed.

NDTR must also be a multiple of the Peripheral burst size multiplied by the peripheral data size, otherwise this could result in a bad DMA behavior.

18.3.10 Double-buffer mode

This mode is available for all the DMA1 and DMA2 streams.

The double-buffer mode is enabled by setting the DBM bit in the DMA_SxCR register.

A double-buffer stream works as a regular (single buffer) stream with the difference that it has two memory pointers. When the double-buffer mode is enabled, the circular mode is automatically enabled (CIRC bit in DMA_SxCR is not relevant) and at each end of transaction, the memory pointers are swapped.

In this mode, the DMA controller swaps from one memory target to another at each end of transaction. This allows the software to process one memory area while the second memory area is being filled/used by the DMA transfer. The double-buffer stream can work in both directions (the memory can be either the source or the destination) as described in Table 103.

*Note: In double-buffer mode, it is possible to update the base address for the AHB memory port on-the-fly (*DMA_SxM0AR or DMA_SxM1AR*) when the stream is enabled, by respecting the following conditions:*

- *When the CT bit is 0 in the DMA_SxCR register, the DMA_SxM1AR register can be written. Attempting to write to this register while CT = 1 sets an error flag (TEIF) and the stream is automatically disabled.*
- *When the CT bit is 1 in the DMA_SxCR register, the DMA_SxM0AR register can be written. Attempting to write to this register while CT = 0, sets an error flag (TEIF) and the stream is automatically disabled.*

To avoid any error condition, it is advised to change the base address as soon as the TCIF flag is asserted because, at this point, the targeted memory must have changed from *memory 0 to 1 (or from 1 to 0) depending on the value of CT in the DMA_SxCR register in accordance with one of the two above conditions.*

For all the other modes (except the double-buffer mode), the memory address registers are write-protected as soon as the stream is enabled.

Table 103. Source and destination address registers in double-buffer mode (DBM = 1)

Bits DIR[1:0] of the DMA_SxCR register	Direction	Source address	Destination address
00	Peripheral-to-memory	DMA_SxPAR	DMA_SxM0AR / DMA_SxM1AR
01	Memory-to-peripheral	DMA_SxM0AR / DMA_SxM1AR	DMA_SxPAR
10	Not allowed(1)		
11	Reserved	-	-

1. When the double-buffer mode is enabled, the circular mode is automatically enabled. Since the memory-to-memory mode is not compatible with the circular mode, when the double-buffer mode is enabled, it is not allowed to configure the memory-to-memory mode.

18.3.11 Programmable data width, packing/unpacking, endianness

The number of data items to be transferred has to be programmed into DMA_SxNDTR (number of data items to transfer bit, NDT) before enabling the stream (except when the flow controller is the peripheral, PFCTRL bit in DMA_SxCR is set).

When using the internal FIFO, the data widths of the source and destination data are programmable through the PSIZE and MSIZE bits in the DMA_SxCR register (can be 8-, 16- or 32-bit).

When PSIZE and MSIZE are not equal:

- The data width of the number of data items to transfer, configured in the DMA_SxNDTR register is equal to the width of the peripheral bus (configured by the PSIZE bits in the DMA_SxCR register). For instance, in case of peripheral-to-memory, memory-to-peripheral or memory-to-memory transfers and if the PSIZE[1:0] bits are configured for half-word, the number of bytes to be transferred is equal to 2 × NDT.
- The DMA controller only copes with little-endian addressing for both source and destination. This is described in *Table 104*.

This packing/unpacking procedure may present a risk of data corruption when the operation is interrupted before the data are completely packed/unpacked. So, to ensure data coherence, the stream may be configured to generate burst transfers: in this case, each group of transfers belonging to a burst are indivisible (refer to *Section 18.3.12*).

In direct mode (DMDIS = 0 in the DMA_SxFCR register), the packing/unpacking of data is not possible. In this case, it is not allowed to have different source and destination transfer data widths: both are equal and defined by the PSIZE bits in the DMA_SxCR register. MSIZE bits are not relevant.

Table 104. Packing/unpacking and endian behavior (bit PINC = MINC = 1)

AHB memory port width	AHB peripheral port width	Number of data items to transfer (NDT)	Memory transfer number	Memory port address / byte lane	Peripheral transfer number	Peripheral port address / byte lane	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
8	16	2	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
8	32	1	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
16	8	4	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
16	16	2	1 2	0x0 / B1|B0[15:0] 0x2 / B1|B0[15:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
16	32	1	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
32	8	4	1	0x0 / B3|B2|B1|B0[31:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
32	16	2	1	0x0 /B3|B2|B1|B0[31:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
32	32	1	1	0x0 /B3|B2|B1|B0 [31:0]	1	0x0 / B3|B2|B1|B0 [31:0]	0x0 / B3|B2|B1|B0[31:0]

Note: Peripheral port may be the source or the destination (it can also be the memory source in the case of memory-to-memory transfer).

PSIZE, MSIZE and NDT[15:0] must be configured so as to ensure that the last transfer is not incomplete. This can occur when the data width of the peripheral port (PSIZE bits) is lower than the data width of the memory port (MSIZE bits). This constraint is summarized in the table below.

Table 105. Restriction on NDT versus PSIZE and MSIZE

PSIZE[1:0] of DMA_SxCR	MSIZE[1:0] of DMA_SxCR	NDT[15:0] of DMA_SxNDTR
00 (8-bit)	01 (16-bit)	Must be a multiple of 2
00 (8-bit)	10 (32-bit)	Must be a multiple of 4
01 (16-bit)	10 (32-bit)	Must be a multiple of 2

18.3.12 Single and burst transfers

The DMA controller can generate single transfers or incremental burst transfers of 4, 8, or 16 beats.

The size of the burst is configured by software independently for the two AHB ports by using the MBURST[1:0] and PBURST[1:0] bits in the DMA_SxCR register.

The burst size indicates the number of beats in the burst, not the number of bytes transferred.

To ensure data coherence, each group of transfers that form a burst are indivisible: AHB transfers are locked and the arbiter of the AHB bus matrix does not degrant the DMA master during the sequence of the burst transfer.

Depending on the single or burst configuration, each DMA request initiates a different number of transfers on the AHB peripheral port:

- When the AHB peripheral port is configured for single transfers, each DMA request generates a data transfer of a byte, half-word or word depending on the PSIZE[1:0] bits in the DMA_SxCR register
- When the AHB peripheral port is configured for burst transfers, each DMA request generates 4,8 or 16 beats of byte, half word or word transfers depending on the PBURST[1:0] and PSIZE[1:0] bits in the DMA_SxCR register.

The same as above has to be considered for the AHB memory port considering the MBURST and MSIZE bits.

In direct mode, the stream can only generate single transfers and the MBURST[1:0] and PBURST[1:0] bits are forced by hardware.

The address pointers (DMA_SxPAR or DMA_SxM0AR registers) must be chosen so as to ensure that all transfers within a burst block are aligned on the address boundary equal to the size of the transfer.

The burst configuration has to be selected in order to respect the AHB protocol, where bursts **must not** cross the 1 Kbyte address boundary because the minimum address space that can be allocated to a single slave is 1 Kbyte. This means that the 1-Kbyte address boundary **must not** be crossed by a burst block transfer, otherwise an AHB error is generated, that is not reported by the DMA registers.

18.3.13 FIFO

FIFO structure

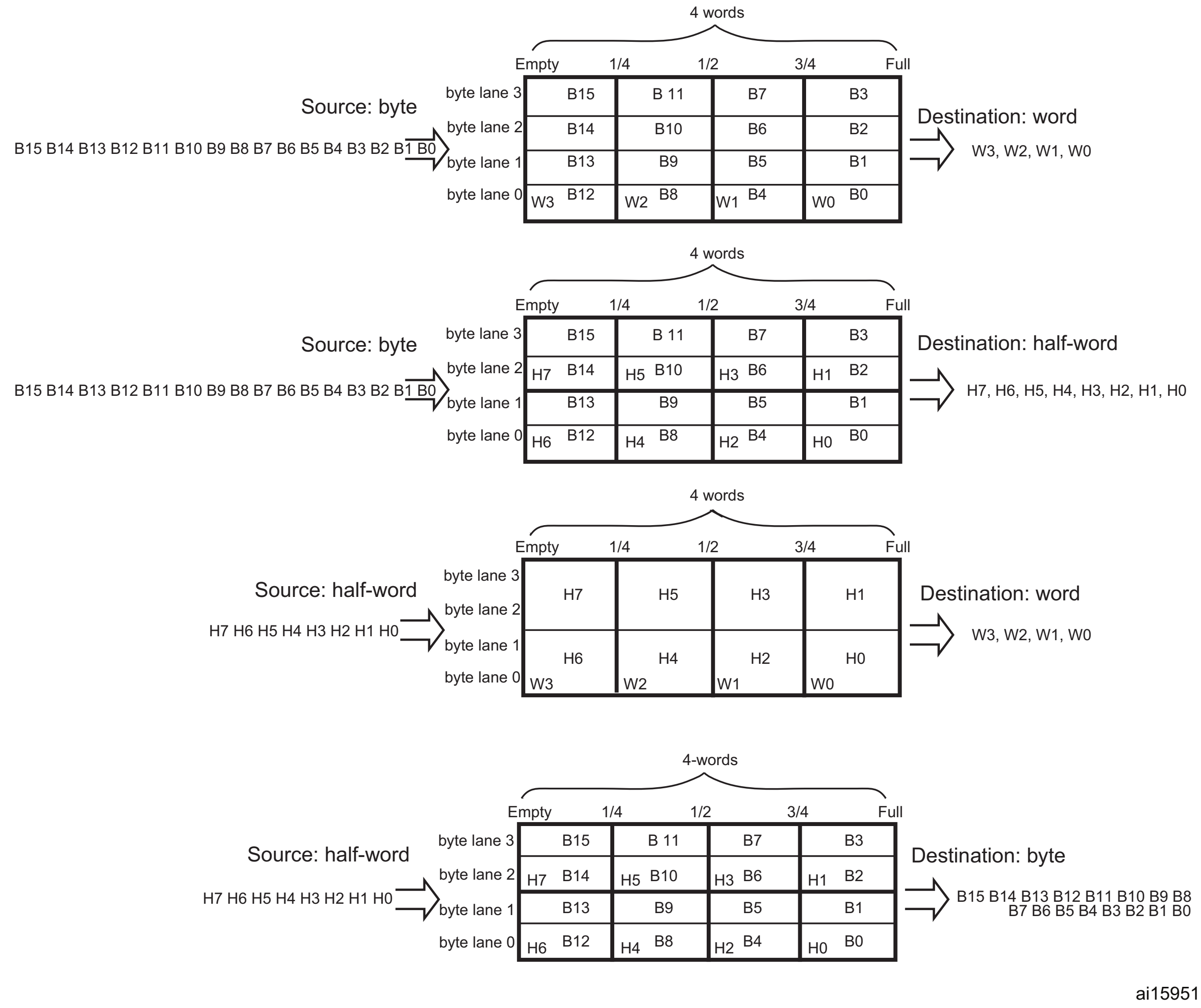
The FIFO is used to temporarily store data coming from the source before transmitting them to the destination.

Each stream has an independent 4-word FIFO and the threshold level is software-configurable between 1/4, 1/2, 3/4 or full.

To enable the use of the FIFO threshold level, the direct mode must be disabled by setting the DMDIS bit in the DMA_SxFCR register.

The structure of the FIFO differs depending on the source and destination data widths, and is described in the figure below.

Figure 132. FIFO structure



FIFO threshold and burst configuration

Caution is required when choosing the FIFO threshold (bits FTH[1:0] of the DMA_SxFCR register) and the size of the memory burst (MBURST[1:0] of the DMA_SxCR register). The content pointed by the FIFO threshold must exactly match an integer number of memory burst transfers. If this is not in the case, a FIFO error (flag FEIFx of the DMA_HISR or DMA_LISR register) is generated when the stream is enabled, then the stream is automatically disabled. The allowed and forbidden configurations are described in the table below. The forbidden configurations are highlighted in gray in the table.

Table 106. FIFO threshold configurations

<table>
<tr><th>MSIZE</th><th>FIFO level</th><th>MBURST = INCR4</th><th>MBURST = INCR8</th><th>MBURST = INCR16</th></tr>
<tr><td rowspan="4">Byte</td><td>1/4</td><td>1 burst of 4 beats</td><td>Forbidden</td><td rowspan="3">Forbidden</td></tr>
<tr><td>1/2</td><td>2 bursts of 4 beats</td><td>1 burst of 8 beats</td></tr>
<tr><td>3/4</td><td>3 bursts of 4 beats</td><td>Forbidden</td></tr>
<tr><td>Full</td><td>4 bursts of 4 beats</td><td>2 bursts of 8 beats</td><td>1 burst of 16 beats</td></tr>
<tr><td rowspan="4">Half-word</td><td>1/4</td><td>Forbidden</td><td rowspan="3">Forbidden</td><td rowspan="8">Forbidden</td></tr>
<tr><td>1/2</td><td>1 burst of 4 beats</td></tr>
<tr><td>3/4</td><td>Forbidden</td></tr>
<tr><td>Full</td><td>2 bursts of 4 beats</td><td>1 burst of 8 beats</td></tr>
<tr><td rowspan="4">Word</td><td>1/4</td><td rowspan="3">Forbidden</td><td rowspan="4">Forbidden</td></tr>
<tr><td>1/2</td></tr>
<tr><td>3/4</td></tr>
<tr><td>Full</td><td>1 burst of 4 beats</td></tr>
</table>

In all cases, the burst size multiplied by the data size must not exceed the FIFO size (data size can be: 1 (byte), 2 (half-word) or 4 (word)).

Incomplete burst transfer at the end of a DMA transfer may happen if one of the following conditions occurs:

- For the AHB peripheral port configuration: the total number of data items (set in the DMA_SxNDTR register) is not a multiple of the burst size multiplied by the data size.
- For the AHB memory port configuration: the number of remaining data items in the FIFO to be transferred to the memory is not a multiple of the burst size multiplied by the data size.

In such cases, the remaining data to be transferred is managed in single mode by the DMA, even if a burst transaction is requested during the DMA stream configuration.

Note: When burst transfers are requested on the peripheral AHB port and the FIFO is used (DMDIS = 1 in the DMA_SxCR register), it is mandatory to respect the following rule to avoid permanent underrun or overrun conditions, depending on the DMA stream direction:

If (PBURST × PSIZE) = FIFO_SIZE (4 words), FIFO_Threshold = 3/4 is forbidden with PSIZE = 1, 2 or 4 and PBURST = 4, 8 or 16.

This rule ensures that enough FIFO space at a time is free to serve the request from the peripheral.

FIFO flush

The FIFO can be flushed when the stream is disabled by resetting the EN bit in the DMA_SxCR register and when the stream is configured to manage peripheral-to-memory or memory-to-memory transfers. If some data are still present in the FIFO when the stream is disabled, the DMA controller continues transferring the remaining data to the destination (even though stream is effectively disabled). When this flush is completed, the transfer complete status bit (TCIFx) in the DMA_LISR or DMA_HISR register is set.

The remaining data counter DMA_SxNDTR keeps the value in this case to indicate how many data items are currently available in the destination memory.

Note that during the FIFO flush operation, if the number of remaining data items in the FIFO to be transferred to memory (in bytes) is less than the memory data width (for example 2 bytes in FIFO while MSIZE is configured to word), data is sent with the data width set in the MSIZE bit in the DMA_SxCR register. This means that memory is written with an undesired value. The software may read the DMA_SxNDTR register to determine the memory area that contains the good data (start address and last address).

If the number of remaining data items in the FIFO is lower than a burst size (if the MBURST bits in DMA_SxCR register are set to configure the stream to manage burst on the AHB memory port), single transactions are generated to complete the FIFO flush.

Direct mode

By default, the FIFO operates in direct mode (DMDIS bit in the DMA_SxFCR is reset) and the FIFO threshold level is not used. This mode is useful when the system requires an immediate and single transfer to or from the memory after each DMA request.

When the DMA is configured in direct mode (FIFO disabled), to transfer data in memory-to-peripheral mode, the DMA preloads one data from the memory to the internal FIFO to ensure an immediate data transfer as soon as a DMA request is triggered by a peripheral.

To avoid saturating the FIFO, it is recommended to configure the corresponding stream with a high priority.

This mode is restricted to transfers where:

- the source and destination transfer widths are equal and both defined by the PSIZE[1:0] bits in DMA_SxCR (MSIZE[1:0] bits are not relevant)
- burst transfers are not possible (PBURST[1:0] and MBURST[1:0] bits in DMA_SxCR are don’t care)

Direct mode must not be used when implementing memory-to-memory transfers.

18.3.14 DMA transfer completion

Different events can generate an end of transfer by setting the TCIFx bit in the DMA_LISR or DMA_HISR status register:

- In DMA flow controller mode:
  – The DMA_SxNDTR counter has reached zero in the memory-to-peripheral mode.
  – The stream is disabled before the end of transfer (by clearing the EN bit in the DMA_SxCR register) and (when transfers are peripheral-to-memory or memory-

    to-memory) all the remaining data have been flushed from the FIFO into the memory.
- In Peripheral flow controller mode:
    - The last external burst or single request has been generated from the peripheral and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory
    - The stream is disabled by software, and (when the DMA is operating in peripheral-to-memory mode) the remaining data have been transferred from the FIFO into the memory

Note: The transfer completion is dependent on the remaining data in FIFO to be transferred into memory only in the case of peripheral-to-memory mode. This condition is not applicable in memory-to-peripheral mode.

If the stream is configured in non-circular mode, after the end of the transfer (that is when the number of data to be transferred reaches zero), the DMA is stopped (EN bit in DMA_SxCR register is cleared by Hardware) and no DMA request is served unless the software reprograms the stream and re-enables it (by setting the EN bit in the DMA_SxCR register).

18.3.15 DMA transfer suspension

At any time, a DMA transfer can be suspended to be restarted later on or to be definitively disabled before the end of the DMA transfer.

There are two cases:

- The stream disables the transfer with no later-on restart from the point where it was stopped. There is no particular action to do, except to clear the EN bit in the DMA_SxCR register to disable the stream. The stream may take time to be disabled (ongoing transfer is completed first). The transfer complete interrupt flag (TCIF in the DMA_LISR or DMA_HISR register) is set in order to indicate the end of transfer. The value of the EN bit in DMA_SxCR is now 0 to confirm the stream interruption. The DMA_SxNDTR register contains the number of remaining data items at the moment when the stream was stopped so that the software can determine how many data items have been transferred before the stream was interrupted.
- The stream suspends the transfer before the number of remaining data items to be transferred in the DMA_SxNDTR register reaches 0. The aim is to restart the transfer later by re-enabling the stream. In order to restart from the point where the transfer was stopped, the software has to read the DMA_SxNDTR register after disabling the stream by writing the EN bit in DMA_SxCR register (and then checking that it is at 0) to know the number of data items already collected. Then:
    - The peripheral and/or memory addresses have to be updated in order to adjust the address pointers
    - The SxNDTR register has to be updated with the remaining number of data items to be transferred (the value read when the stream was disabled)
    - The stream may then be re-enabled to restart the transfer from the point it was stopped

Note: A transfer complete interrupt flag (TCIF in DMA_LISR or DMA_HISR) is set to indicate the end of transfer due to the stream interruption.

18.3.16 Flow controller

The entity that controls the number of data to be transferred is known as the flow controller. This flow controller is configured independently for each stream using the PFCTRL bit in the DMA_SxCR register.

The flow controller can be:

- The DMA controller: in this case, the number of data items to be transferred is programmed by software into the DMA_SxNDTR register before the DMA stream is enabled.
- The peripheral source or destination: this is the case when the number of data items to be transferred is unknown. The peripheral indicates by hardware to the DMA controller when the last data are being transferred. This feature is only supported for peripherals that are able to signal the end of the transfer.

When the peripheral flow controller is used for a given stream, the value written into the DMA_SxNDTR has no effect on the DMA transfer. Actually, whatever the value written, it is forced by hardware to 0xFFFF as soon as the stream is enabled, to respect the following schemes:

- Anticipated stream interruption: EN bit in DMA_SxCR register is reset to 0 by the software to stop the stream before the last data hardware signal (single or burst) is sent by the peripheral. In such a case, the stream is switched off and the FIFO flush is triggered in the case of a peripheral-to-memory DMA transfer. The TCIFx flag of the corresponding stream is set in the status register to indicate the DMA completion. To know the number of data items transferred during the DMA transfer, read the DMA_SxNDTR register and apply the following formula:
  - Number_of_data_transferred = 0xFFFF – DMA_SxNDTR
- Normal stream interruption due to the reception of a last data hardware signal: the stream is automatically interrupted when the peripheral requests the last transfer (single or burst) and when this transfer is complete. the TCIFx flag of the corresponding stream is set in the status register to indicate the DMA transfer completion. To know the number of data items transferred, read the DMA_SxNDTR register and apply the same formula as above.
- The DMA_SxNDTR register reaches 0: the TCIFx flag of the corresponding stream is set in the status register to indicate the forced DMA transfer completion. The stream is automatically switched off even though the last data hardware signal (single or burst) has not been yet asserted. The already transferred data is not lost. This means that a maximum of 65535 data items can be managed by the DMA in a single transaction, even in peripheral flow control mode.

Note: When configured in memory-to-memory mode, the DMA is always the flow controller and the PFCTRL bit is forced to 0 by hardware.

The circular mode is forbidden in the peripheral flow controller mode.

18.3.17 Summary of the possible DMA configurations

The table below summarizes the different possible DMA configurations. The forbidden configurations are highlighted in gray in the table.

Table 107. Possible DMA configurations

DMA transfer mode	Source	Destination	Flow controller	Circular mode	Transfer type	Direct mode	Double-buffer mode
Peripheral-to-memory	AHB peripheral port	AHB memory port	DMA	Possible	single	Possible	Possible
					burst	Forbidden	
			Peripheral	Forbidden	single	Possible	Forbidden
					burst	Forbidden	
Memory-to-peripheral	AHB memory port	AHB peripheral port	DMA	Possible	single	Possible	Possible
					burst	Forbidden	
			Peripheral	Forbidden	single	Possible	Forbidden
					burst	Forbidden	
Memory-to-memory	AHB peripheral port	AHB memory port	DMA only	Forbidden	single	Forbidden	Forbidden
					burst		

18.3.18 Stream configuration procedure

The following sequence must be followed to configure a DMA stream x (where x is the stream number):

1. If the stream is enabled, disable it by resetting the EN bit in the DMA_SxCR register, then read this bit in order to confirm that there is no ongoing stream operation. Writing this bit to 0 is not immediately effective since it is actually written to 0 once all the current transfers are finished. When the EN bit is read as 0, this means that the stream is ready to be configured. It is therefore necessary to wait for the EN bit to be cleared before starting any stream configuration. All the stream dedicated bits set in the status register (DMA_LISR and DMA_HISR) from the previous data block DMA transfer must be cleared before the stream can be re-enabled.
2. Set the peripheral port register address in the DMA_SxPAR register. The data is moved from/ to this address to/ from the peripheral port after the peripheral event.
3. Set the memory address in the DMA_SxMA0R register (and in the DMA_SxMA1R register in the case of a double-buffer mode). The data is written to or read from this memory after the peripheral event.
4. Configure the total number of data items to be transferred in the DMA_SxNDTR register. After each peripheral event or each beat of the burst, this value is decremented.
5. Use DMAMUX to route a DMA request line to the DMA channel.
6. If the peripheral is intended to be the flow controller and if it supports this feature, set the PFCTRL bit in the DMA_SxCR register.
7. Configure the stream priority using the PL[1:0] bits in the DMA_SxCR register.
8. Configure the FIFO usage (enable or disable, threshold in transmission and reception)

9. Configure the data transfer direction, peripheral and memory incremented/fixed mode, single or burst transactions, peripheral and memory data widths, circular mode, double-buffer mode and interrupts after half and/or full transfer, and/or errors in the DMA_SxCR register.
10. Activate the stream by setting the EN bit in the DMA_SxCR register.

As soon as the stream is enabled, it can serve any DMA request from the peripheral connected to the stream.

Once half the data have been transferred on the AHB destination port, the half-transfer flag (HTIF) is set and an interrupt is generated if the half-transfer interrupt enable bit (HTIE) is set. At the end of the transfer, the transfer complete flag (TCIF) is set and an interrupt is generated if the transfer complete interrupt enable bit (TCIE) is set.

Warning: To switch off a peripheral connected to a DMA stream request, it is mandatory to, first, switch off the DMA stream to which the peripheral is connected, then to wait for EN bit = 0. Only then can the peripheral be safely disabled.

18.3.19 Error management

The DMA controller can detect the following errors:

- **Transfer error**: the transfer error interrupt flag (TEIFx) is set when:
  - a bus error occurs during a DMA read or a write access
  - a write access is requested by software on a memory address register in double-buffer mode whereas the stream is enabled and the current target memory is the one impacted by the write into the memory address register (refer to *Section 18.3.10: Double-buffer mode*)
- **FIFO error**: the FIFO error interrupt flag (FEIFx) is set if:
  - a FIFO underrun condition is detected
  - a FIFO overrun condition is detected (no detection in memory-to-memory mode because requests and transfers are internally managed by the DMA)
  - the stream is enabled while the FIFO threshold level is not compatible with the size of the memory burst (refer to *Table 106: FIFO threshold configurations*)
- **Direct mode error**: the direct mode error interrupt flag (DMEIFx) can only be set in the peripheral-to-memory mode while operating in direct mode and when the MINC bit in the DMA_SxCR register is cleared. This flag is set when a DMA request occurs while the previous data have not yet been fully transferred into the memory (because the memory bus was not granted). In this case, the flag indicates that two data items were be transferred successively to the same destination address, which could be an issue if the destination is not able to manage this situation

In direct mode, the FIFO error flag can also be set under the following conditions:

- In the peripheral-to-memory mode, the FIFO can be saturated (overrun) if the memory bus is not granted for several peripheral requests.
- In the memory-to-peripheral mode, an underrun condition may occur if the memory bus has not been granted before a peripheral request occurs.

If the TEIFx or the FEIFx flag is set due to incompatibility between burst size and FIFO threshold level, the faulty stream is automatically disabled through a hardware clear of its EN bit in the corresponding stream configuration register (DMA_SxCR).

If the DMEIFx or the FEIFx flag is set due to an overrun or underrun condition, the faulty stream is not automatically disabled and it is up to the software to disable or not the stream by resetting the EN bit in the DMA_SxCR register. This is because there is no data loss when this kind of errors occur.

When the stream's error interrupt flag (TEIF, FEIF, DMEIF) in the DMA_LISR or DMA_HISR register is set, an interrupt is generated if the corresponding interrupt enable bit (TEIE, FEIE, DMEIE) in the DMA_SxCR or DMA_SxFCR register is set.

Note: When a FIFO overrun or underrun condition occurs, the data is not lost because the peripheral request is not acknowledged by the stream until the overrun or underrun condition is cleared. If this acknowledge takes too much time, the peripheral itself may detect an overrun or underrun condition of its internal buffer and data might be lost.

18.4 DMA interrupts

For each DMA stream, an interrupt can be produced on the following events:

- Half-transfer reached
- Transfer complete
- Transfer error
- FIFO error (overrun, underrun or FIFO level error)
- Direct mode error

Separate interrupt enable control bits are available for flexibility as shown in the table below.

Table 108. DMA interrupt requests

Interrupt event	Event flag	Enable control bit
Half-transfer	HTIF	HTIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE
FIFO overrun/underrun	FEIF	FEIE
Direct mode error	DMEIF	DMEIE

Note: Before setting an enable control bit EN = 1, the corresponding event flag must be cleared, otherwise an interrupt is immediately generated.

18.5 DMA registers

The DMA registers have to be accessed by words (32 bits).

18.5.1 DMA low interrupt status register (DMA_LISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2
				r	r	r	r		r	r	r	r	r		r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[3:0]**: Stream x transfer complete interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_LIFCR register.
0: No transfer complete event on stream x
1: A transfer complete event occurred on stream x.

Bits 26, 20, 10, 4 **HTIF[3:0]**: Stream x half transfer interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_LIFCR register.
0: No half transfer event on stream x
1: An half transfer event occurred on stream x.

Bits 25, 19, 9, 3 **TEIF[3:0]**: Stream x transfer error interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_LIFCR register.
0: No transfer error on stream x
1: A transfer error occurred on stream x.

Bits 24, 18, 8, 2 **DMEIF[3:0]**: Stream x direct mode error interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_LIFCR register.
0: No direct mode error on stream x
1: A direct mode error occurred on stream x.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIF[3:0]**: Stream x FIFO error interrupt flag (x = 3 to 0)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_LIFCR register.
0: No FIFO error event on stream x
1: A FIFO error event occurred on stream x.

18.5.2 DMA high interrupt status register (DMA_HISR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TCIF7	HTIF7	TEIF7	DMEIF 7	Res.	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF 6	Res.	FEIF6
				r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TCIF5	HTIF5	TEIF5	DMEIF 5	Res.	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF 4	Res.	FEIF4
				r	r	r	r		r	r	r	r	r		r

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **TCIF[7:4]**: Stream x transfer complete interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_HIFCR register.
0: No transfer complete event on stream x
1: A transfer complete event occurred on stream x.

Bits 26, 20, 10, 4 **HTIF[7:4]**: Stream x half transfer interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_HIFCR register.
0: No half transfer event on stream x
1: An half transfer event occurred on stream x.

Bits 25, 19, 9, 3 **TEIF[7:4]**: Stream x transfer error interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_HIFCR register.
0: No transfer error on stream x
1: A transfer error occurred on stream x.

Bits 24, 18, 8, 2 **DMEIF[7:4]**: Stream x direct mode error interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_HIFCR register.
0: No direct mode error on stream x
1: A direct mode error occurred on stream x.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **FEIF[7:4]**: Stream x FIFO error interrupt flag (x = 7 to 4)
This bit is set by hardware. It is cleared by software writing 1 to the corresponding bit in DMA_HIFCR register.
0: No FIFO error event on stream x
1: A FIFO error event occurred on stream x.

18.5.3 DMA low interrupt flag clear register (DMA_LIFCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2
				w	w	w	w		w	w	w	w	w		w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[3:0]**: Stream x clear transfer complete interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_LISR register.

Bits 26, 20, 10, 4 **CHTIF[3:0]**: Stream x clear half transfer interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_LISR register

Bits 25, 19, 9, 3 **CTEIF[3:0]**: Stream x clear transfer error interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_LISR register.

Bits 24, 18, 8, 2 **CDMEIF[3:0]**: Stream x clear direct mode error interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_LISR register.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIF[3:0]**: Stream x clear FIFO error interrupt flag (x = 3 to 0)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_LISR register.

18.5.4 DMA high interrupt flag clear register (DMA_HIFCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Res.	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Res.	CFEIF6
				w	w	w	w		w	w	w	w	w		w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Res.	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Res.	CFEIF4
				w	w	w	w		w	w	w	w	w		w

Bits 31:28, 15:12 Reserved, must be kept at reset value.

Bits 27, 21, 11, 5 **CTCIF[7:4]**: Stream x clear transfer complete interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding TCIFx flag in the DMA_HISR register.

Bits 26, 20, 10, 4 **CHTIF[7:4]**: Stream x clear half transfer interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding HTIFx flag in the DMA_HISR register.

Bits 25, 19, 9, 3 **CTEIF[7:4]**: Stream x clear transfer error interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding TEIFx flag in the DMA_HISR register.

Bits 24, 18, 8, 2 **CDMEIF[7:4]**: Stream x clear direct mode error interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding DMEIFx flag in the DMA_HISR register.

Bits 23, 17, 7, 1 Reserved, must be kept at reset value.

Bits 22, 16, 6, 0 **CFEIF[7:4]**: Stream x clear FIFO error interrupt flag (x = 7 to 4)
Writing 1 to this bit clears the corresponding CFEIFx flag in the DMA_HISR register.

18.5.5 DMA stream x configuration register (DMA_SxCR)

This register is used to configure the concerned stream.

Address offset: 0x010 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST[1:0]		PBURST[1:0]		TRBUFF	CT	DBM	PL[1:0]	
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:23 **MBURST[1:0]**: Memory burst transfer configuration
These bits are set and cleared by software.
00: Single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN = 0.
In direct mode, these bits are forced to 0x0 by hardware as soon as bit EN = 1.

Bits 22:21 **PBURST[1:0]**: Peripheral burst transfer configuration
These bits are set and cleared by software.
00: Single transfer
01: INCR4 (incremental burst of 4 beats)
10: INCR8 (incremental burst of 8 beats)
11: INCR16 (incremental burst of 16 beats)
These bits are protected and can be written only if EN = 0.
In direct mode, these bits are forced to 0x0 by hardware.

Bit 20 **TRBUFF**: Enable the DMA to handle bufferable transfers
0: Bufferable transfers not enabled
1: Bufferable transfers enabled

Note: This bit must be set to 1 if the DMA stream manages UART/USART transfers.

Bit 19 **CT**: Current target (only in double-buffer mode)
This bit is set and cleared by hardware. It can also be written by software.
0: Current target memory is memory 0 (addressed by the DMA_SxM0AR pointer).
1: Current target memory is memory 1 (addressed by the DMA_SxM1AR pointer).
This bit can be written only if EN = 0 to indicate the target memory area of the first transfer. Once the stream is enabled, this bit operates as a status flag indicating which memory area is the current target.

Bit 18 **DBM**: Double-buffer mode
This bit is set and cleared by software.
0: No buffer switching at the end of transfer
1: Memory target switched at the end of the DMA transfer
This bit is protected and can be written only if EN = 0.

Bits 17:16 **PL[1:0]**: priority level
These bits are set and cleared by software.
00: Low
01: Medium
10: High
11: Very high
These bits are protected and can be written only if EN = 0.

Bit 15 **PINCOS**: Peripheral increment offset size
This bit is set and cleared by software
0: The offset size for the peripheral address calculation is linked to the PSIZE.
1: The offset size for the peripheral address calculation is fixed to 4 (32-bit alignment).
This bit has no meaning if bit PINC = 0.
This bit is protected and can be written only if EN = 0.
This bit is forced low by hardware when the stream is enabled (EN = 1) if the direct mode is selected or if PBURST are different from 00.

Bits 14:13 **MSIZE[1:0]**: Memory data size
These bits are set and cleared by software.
00: Byte (8-bit)
01: Half-word (16-bit)
10: Word (32-bit)
11: Reserved
These bits are protected and can be written only if EN = 0.
In direct mode, MSIZE is forced by hardware to the same value as PSIZE as soon as EN = 1.

Bits 12:11 **PSIZE[1:0]**: Peripheral data size
These bits are set and cleared by software.
00: Byte (8-bit)
01: Half-word (16-bit)
10: Word (32-bit)
11: Reserved
These bits are protected and can be written only if EN = 0.

Bit 10 **MINC**: Memory increment mode
This bit is set and cleared by software.
0: Memory address pointer fixed
1: Memory address pointer incremented after each data transfer (increment is done according to MSIZE)
This bit is protected and can be written only if EN = 0.

Bit 9 **PINC**: Peripheral increment mode

This bit is set and cleared by software.
0: Peripheral address pointer fixed
1: Peripheral address pointer incremented after each data transfer (increment done according to PSIZE)
This bit is protected and can be written only if EN = 0.

Bit 8 **CIRC**: Circular mode

This bit is set and cleared by software and can be cleared by hardware.
0: Circular mode disabled
1: Circular mode enabled
When the peripheral is the flow controller (bit PFCTRL = 1), and the stream is enabled (EN = 1), then this bit is automatically forced by hardware to 0.
It is automatically forced by hardware to 1 if the DBM bit is set, as soon as the stream is enabled (EN = 1).

Bits 7:6 **DIR[1:0]**: Data transfer direction

These bits are set and cleared by software.
00: Peripheral-to-memory
01: Memory-to-peripheral
10: Memory-to-memory
11: Reserved
These bits are protected and can be written only if EN = 0.

Bit 5 **PFCTRL**: Peripheral flow controller

This bit is set and cleared by software.
0: DMA is the flow controller.
1: The peripheral is the flow controller.
This bit is protected and can be written only if EN = 0.
When the memory-to-memory mode is selected (bits DIR[1:0] = 10), then this bit is automatically forced to 0 by hardware.

Bit 4 **TCIE**: Transfer complete interrupt enable

This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 3 **HTIE**: Half transfer interrupt enable

This bit is set and cleared by software.
0: HT interrupt disabled
1: HT interrupt enabled

Bit 2 **TEIE**: Transfer error interrupt enable

This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 1 **DMEIE**: Direct mode error interrupt enable

This bit is set and cleared by software.
0: DME interrupt disabled
1: DME interrupt enabled

Bit 0 **EN**: Stream enable/flag stream ready when read low
This bit is set and cleared by software.
0: Stream disabled
1: Stream enabled
This bit is cleared by hardware:
– on a DMA end of transfer (stream ready to be configured)
– if a transfer error occurs on the AHB master buses
– when the FIFO threshold on memory AHB port is not compatible with the burst size
When this bit is read as 0, the software is allowed to program the configuration and FIFO bits registers. It is forbidden to write these registers when the EN bit is read as 1.
Note: Before setting EN bit to 1 to start a new transfer, the event flags corresponding to the stream in DMA_LISR or DMA_HISR register must be cleared.

18.5.6 DMA stream x number of data register (DMA_SxNDTR)

Address offset: 0x014 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **NDT[15:0]**: Number of data items to transfer (0 up to 65535)
This bitfield can be written only when the stream is disabled. When the stream is enabled, this bitfield is read-only, indicating the remaining data items to be transmitted. This bitfield decrements after each DMA transfer.
Once the transfer is completed, this bitfield can either stay at zero (when the stream is in normal mode), or be reloaded automatically with the previously programmed value in the following cases:
– when the stream is configured in circular mode
– when the stream is enabled again by setting EN bit to 1
If the value of this bitfield is zero, no transaction can be served even if the stream is enabled.

18.5.7 DMA stream x peripheral address register (DMA_SxPAR)

Address offset: 0x018 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PAR[31:0]**: Peripheral address
Base address of the peripheral data register from/to which the data is read/written.
These bits are write-protected and can be written only when bit EN = 0 in DMA_SxCR.

18.5.8 DMA stream x memory 0 address register (DMA_SxM0AR)

Address offset: 0x01C + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M0A[31:0]**: Memory 0 address
Base address of memory area 0 from/to which the data is read/written.
These bits are write-protected. They can be written only if:
- the stream is disabled (EN = 0 in DMA_SxCR) or
- the stream is enabled (EN = 1 in DMA_SxCR) and CT = 1 in DMA_SxCR (in double-buffer mode).

18.5.9 DMA stream x memory 1 address register (DMA_SxM1AR)

Address offset: 0x020 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **M1A[31:0]**: Memory 1 address (used in case of double-buffer mode)
Base address of memory area 1 from/to which the data is read/written.
This bitfield is used only for the double-buffer mode.
These bits are write-protected. They can be written only if:
- the stream is disabled (EN = 0 in DMA_SxCR) or
- the stream is enabled (EN = 1 in DMA_SxCR) and bit CT = 0 in DMA_SxCR .

18.5.10 DMA stream x FIFO control register (DMA_SxFCR)

Address offset: 0x024 + 0x18 * x, (x = 0 to 7)

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEIE	Res.	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **FEIE**: FIFO error interrupt enable
This bit is set and cleared by software.
0: FE interrupt disabled
1: FE interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bits 5:3 **FS[2:0]**: FIFO status
These bits are read-only.
000: 0 < fifo_level < 1/4
001: 1/4 ≤ fifo_level < 1/2
010: 1/2 ≤ fifo_level < 3/4
011: 3/4 ≤ fifo_level < full
100: FIFO is empty.
101: FIFO is full.
Others: Reserved (no meaning)
These bits are not relevant in the direct mode (DMDIS = 0).

Bit 2 **DMDIS**: Direct mode disable
This bit is set and cleared by software. It can be set by hardware.
0: Direct mode enabled
1: Direct mode disabled
This bit is protected and can be written only if EN = 0.
This bit is set by hardware if the memory-to-memory mode is selected (DIR = 10 in DMA_SxCR), and EN = 1 in DMA_SxCR because the direct mode is not allowed in the memory-to-memory configuration.

Bits 1:0 **FTH[1:0]**: FIFO threshold selection
These bits are set and cleared by software.
00: 1/4 full FIFO
01: 1/2 full FIFO
10: 3/4 full FIFO
11: Full FIFO
These bits are not used in the direct mode when the DMIS = 0.
These bits are protected and can be written only if EN = 0.

18.5.11 DMA hardware configuration 2 register (DMA_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CHSEL_WIDTH[2:0]			Res.	Res.	Res.	WRITE_BUFFERABLE	Res.	Res.	FIFO_SIZE[1:0]	
					r	r	r				r			r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **CHSEL_WIDTH[2:0]**: Width of the CHSEL field of any DMA_SxCR register common to all streams

In [0..6] range:
0: No programmable selection
1: 2 channels programmable selection
2: Up to 8 channels programmable selection
3: Up to 16 channels programmable selection
4: Up to 32 channels programmable selection
5: Up to 64 channels programmable selection
6: Up to 128 channels programmable selection

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **WRITE_BUFFERABLE**:

0: DMA acknowledge signal is de-asserted one cycle after occurred both 1) the last cycle of the data phase of the bus access on its AHB peripheral master port and 2) the de-assertion of the peripheral request
1: DMA acknowledge signal is de-asserted exactly one cycle after the last cycle of the data phase of the bus access on its AHB peripheral master port
In any case, DMA acknowledge signal is asserted one cycle after the address phase of the bus access on its AHB peripheral master port.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **FIFO_SIZE[1:0]**: FIFO size, common to all streams

In [0..3] range:
0: 2-word FIFO
1: 4-word FIFO
2: 8-word FIFO
3: 16-word FIFO

18.5.12 DMA hardware configuration 1 register (DMA_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x2222 2222

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DMA_DEF7[1:0]		Res.	Res.	DMA_DEF6[1:0]		Res.	Res.	DMA_DEF5[1:0]		Res.	Res.	DMA_DEF4[1:0]	
		r	r			r	r			r	r			r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DMA_DEF3[1:0]		Res.	Res.	DMA_DEF2[1:0]		Res.	Res.	DMA_DEF1[1:0]		Res.	Res.	DMA_DEF0[1:0]	
		r	r			r	r			r	r			r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **DMA_DEF7[1:0]**: Stream 7 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **DMA_DEF6[1:0]**: Stream 6 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **DMA_DEF5[1:0]**: Stream 5 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **DMA_DEF4[1:0]**: Stream 4 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **DMA_DEF3[1:0]**: Stream 3 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **DMA_DEF2[1:0]**: Stream 2 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DMA_DEF1[1:0]**: Stream 1 type
0: None
1: Regular
2: Double-buffer
3: Reserved

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DMA_DEF0[1:0]**: Stream 0 type
0: None
1: Regular
2: Double-buffer
3: Reserved

18.5.13 DMA version register (DMA_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: DMA major revision

Bits 3:0 **MINREV[3:0]**: DMA minor revision

18.5.14 DMA identification register (DMA_IPDR)

Address offset: 0x3F8

Reset value: 0x0010 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: DMA size identification

18.5.15 DMA size identification register (DMA_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: size identification

This register identifies the DMA as a peripheral with a size of 1 Kbyte address space.

18.5.16 DMA register map

Table 109. DMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA_LISR	Res.	Res.	Res.	Res.	TCIF3	HTIF3	TEIF3	DMEIF3	Res.	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Res.	FEIF2	Res.	Res.	Res.	Res.	TCIF1	HTIF1	TEIF1	DMEIF1	Res.	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Res.	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x004	DMA_HISR	Res.	Res.	Res.	Res.	TCIF7	HTIF7	TEIF7	DMEIF7	Res.	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Res.	FEIF6	Res.	Res.	Res.	Res.	TCIF5	HTIF5	TEIF5	DMEIF5	Res.	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Res.	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x008	DMA_LIFCR	Res.	Res.	Res.	Res.	CTCIF3	CHTIF3	TEIF3	CDMEIF3	Res.	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Res.	CFEIF2	Res.	Res.	Res.	Res.	CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Res.	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Res.	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x00C	DMA_HIFCR	Res.	Res.	Res.	Res.	CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Res.	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6	Res.	CFEIF6	Res.	Res.	Res.	Res.	CTCIF5	CHTIF5	CTEIF5	CDMEIF5	Res.	CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4	Res.	CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x010	DMA_S0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST [1:0]		PBURST [1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA_S0NDTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	DMA_S0PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	DMA_S0M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DMA_S0M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	DMA_S0FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x028	DMA_S1CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	DMA_S1NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	DMA_S1PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA_S1M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA_S1M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA_S1FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x040	DMA_S2CR	Res	Res	Res	Res	Res	Res	Res	MBURST[1:0]		PBURST[1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	DMA_S2NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA_S2PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA_S2M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	DMA_S2M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	DMA_S2FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x058	DMA_S3CR	Res	Res	Res	Res	Res	Res	Res	MBURST [1:0]		PBURST [1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x05C	DMA_S3NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DMA_S3PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	DMA_S3M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	DMA_S3M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	DMA_S3FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x070	DMA_S4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MBURST [1:0]		PBURST [1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	DMA_S4NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	DMA_S4PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	DMA_S4M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	DMA_S4M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMA_S4FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x088	DMA_S5CR	Res	Res	Res	Res	Res	Res	Res	MBURST [1:0]		PBURST [1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DMA_S5NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	DMA_S5PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	DMA_S5M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x098	DMA_S5M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	DMA_S5FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0A0	DMA_S6CR	Res	Res	Res	Res	Res	Res	Res	MBURST [1:0]		PBURST [1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	DMA_S6NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	DMA_S6PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	DMA_S6M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0	DMA_S6M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	DMA_S6FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0B8	DMA_S7CR	Res	Res	Res	Res	Res	Res	Res	MBURST [1:0]		PBURST [1:0]		TRBUFF	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	DMA_S7NDTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	NDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0	DMA_S7PAR	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	DMA_S7M0AR	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8	DMA_S7M1AR	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC	DMA_S7FCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FEIE	Res	FS[2:0]			DMDIS	FTH[1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0D0-0x3D8	Reserved	Reserved																															

Table 109. DMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3EC	DMA_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHSEL_WIDTH[2:0]			Res.	Res.	Res.	WRITE_BUFFERABLE	Res.	Res.	FIFO_SIZE[1:0]	
	Reset value																						0	0	0				0			0	1
0x3F0	DMA_HWCFGR1	Res.	Res.	DMA_DEF7		Res.	Res.	DMA_DEF6		Res.	Res.	DMA_DEF5		Res.	Res.	DMA_DEF4		Res.	Res.	DMA_DEF3		Res.	Res.	DMA_DEF2		Res.	Res.	DMA_DEF1		Res.	Res.	DMA_DEF0	
	Reset value			1	0			1	0			1	0			1	0			1	0			1	0			1	0			1	0
0x3F4	DMA_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0x3F8	DMA_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	DMA_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

19 DMA request multiplexer (DMAMUX)

19.1 Introduction

A peripheral indicates a request for DMA transfer by setting its DMA request signal. The DMA request is pending until served by the DMA controller that generates a DMA acknowledge signal, and the corresponding DMA request signal is deasserted.

In this document, the set of control signals required for the DMA request/acknowledge protocol is not explicitly shown or described, and it is referred to as DMA request line.

The DMAMUX request multiplexer enables routing a DMA request line between the peripherals and the DMA controllers of the product. The routing function is ensured by a programmable multi-channel DMA request line multiplexer. Each channel selects a unique DMA request line, unconditionally or synchronously with events from its DMAMUX synchronization inputs. The DMAMUX may also be used as a DMA request generator from programmable events on its input trigger signals.

The number of DMAMUX instances and their main characteristics are specified in *Section 19.3.1*.

The assignment of DMAMUX request multiplexer inputs to the DMA request lines from peripherals and to the DMAMUX request generator outputs, the assignment of DMAMUX request multiplexer outputs to DMA controller channels, and the assignment of DMAMUX synchronizations and trigger inputs to internal and external signals depend upon product implementation. They are detailed in *Section 19.3.2*.

19.2 DMAMUX main features

- 16-channel programmable DMA request line multiplexer output
- 8-channel DMA request generator
- 8 trigger inputs to DMA request generator
- 8 synchronization inputs
- Per DMA request generator channel:
  - DMA request trigger input selector
  - DMA request counter
  - Event overrun flag for selected DMA request trigger input
- Per DMA request line multiplexer channel output:
  - 108 input DMA request lines from peripherals
  - One DMA request line output
  - Synchronization input selector
  - DMA request counter
  - Event overrun flag for selected synchronization input
  - One event output, for DMA request chaining
- Identification registers, with associated IP revision and hardware configuration.

19.3 DMAMUX implementation

19.3.1 DMAMUX instantiation

DMAMUX is instantiated with the hardware configuration parameters listed in the following table.

Table 110. DMAMUX instantiation

Feature	DMAMUX
Number of DMAMUX output request channels	16
Number of DMAMUX request generator channels	8
Number of DMAMUX request trigger inputs	8
Number of DMAMUX synchronization inputs	8
Number of DMAMUX peripheral request inputs	108

19.3.2 DMAMUX mapping

The mapping of resources to DMAMUX is hardwired.

DMAMUX is used with DMA1 and DMA2:

- DMAMUX channels 0 to 7 are connected to DMA1 channels 0 to 7
- DMAMUX channels 8 to 15 are connected to DMA2 channels 0 to 7

Table 111. DMAMUX: assignment of multiplexer inputs to resources

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
1	dmamux_req_gen0	44	USART2_TX	87	SAI1_A
2	dmamux_req_gen1	45	USART3_RX	88	SAI1_B
3	dmamux_req_gen2	46	USART3_TX	89	SAI2_A
4	dmamux_req_gen3	47	TIM8_CH1	90	SAI2_B
5	dmamux_req_gen4	48	TIM8_CH2	91	DFSDM1_FLT4
6	dmamux_req_gen5	49	TIM8_CH3	92	DFSDM1_FLT5
7	dmamux_req_gen6	50	TIM8_CH4	93	SPDIFRX_DT
8	dmamux_req_gen7	51	TIM8_UP	94	SPDIFRX_CS
9	ADC1	52	TIM8_TRIG	95	-
10	ADC2	53	TIM8_COM	96	-
11	TIM1_CH1	54	-	97	-
12	TIM1_CH2	55	TIM5_CH1	98	-
13	TIM1_CH3	56	TIM5_CH2	99	SAI4_A
14	TIM1_CH4	57	TIM5_CH3	100	SAI4_B
15	TIM1_UP	58	TIM5_CH4	101	DFSDM1_FLT0
16	TIM1_TRIG	59	TIM5_UP	102	DFSDM1_FLT1
17	TIM1_COM	60	TIM5_TRIG	103	DFSDM1_FLT2
18	TIM2_CH1	61	SPI3_RX	104	DFSDM1_FLT3
19	TIM2_CH2	62	SPI3_TX	105	TIM15_CH1
20	TIM2_CH3	63	UART4_RX	106	TIM15_UP
21	TIM2_CH4	64	UART4_TX	107	TIM15_TRIG
22	TIM2_UP	65	UART5_RX	108	TIM15_COM
23	TIM3_CH1	66	UART5_TX	109	TIM16_CH1
24	TIM3_CH2	67	DAC1	110	TIM16_UP
25	TIM3_CH3	68	DAC2	111	TIM17_CH1
26	TIM3_CH4	69	TIM6_UP	112	TIM17_UP
27	TIM3_UP	70	TIM7_UP	113	SAI3_A
28	TIM3_TRIG	71	USART6_RX	114	SAI3_B
29	TIM4_CH1	72	USART6_TX	115	I2C5_RX
30	TIM4_CH2	73	I2C3_RX	116	I2C5_TX
31	TIM4_CH3	74	I2C3_TX	117	Reserved
32	TIM4_UP	75	DCMI	118	Reserved
33	I2C1_RX	76	CRYP2_IN	119	Reserved
34	I2C1_TX	77	CRYP2_OUT	120	Reserved
35	I2C2_RX	78	HASH2_IN	121	Reserved
36	I2C2_TX	79	UART7_RX	122	Reserved

Table 111. DMAMUX: assignment of multiplexer inputs to resources (continued)

DMA request MUX input	Resource	DMA request MUX input	Resource	DMA request MUX input	Resource
37	SPI1_RX	80	UART7_TX	123	Reserved
38	SPI1_TX	81	UART8_RX	124	Reserved
39	SPI2_RX	82	UART8_TX	125	Reserved
40	SPI2_TX	83	SPI4_RX	126	Reserved
41	-	84	SPI4_TX	127	Reserved
42	-	85	SPI5_RX	-	-
43	USART2_RX	86	SPI5_TX	-	-

Table 112. DMAMUX: assignment of trigger inputs to resources

Trigger input	Resource	Trigger input	Resource
0	dmamux_evt0	5	LPTIMER2_OUT
1	dmamux_evt1	6	LPTIMER3_OUT
2	dmamux_evt2	7	extit0
3	LPTIMER1_OUT	8	TIM12_TRGO

Table 113. DMAMUX: assignment of synchronization inputs to resources

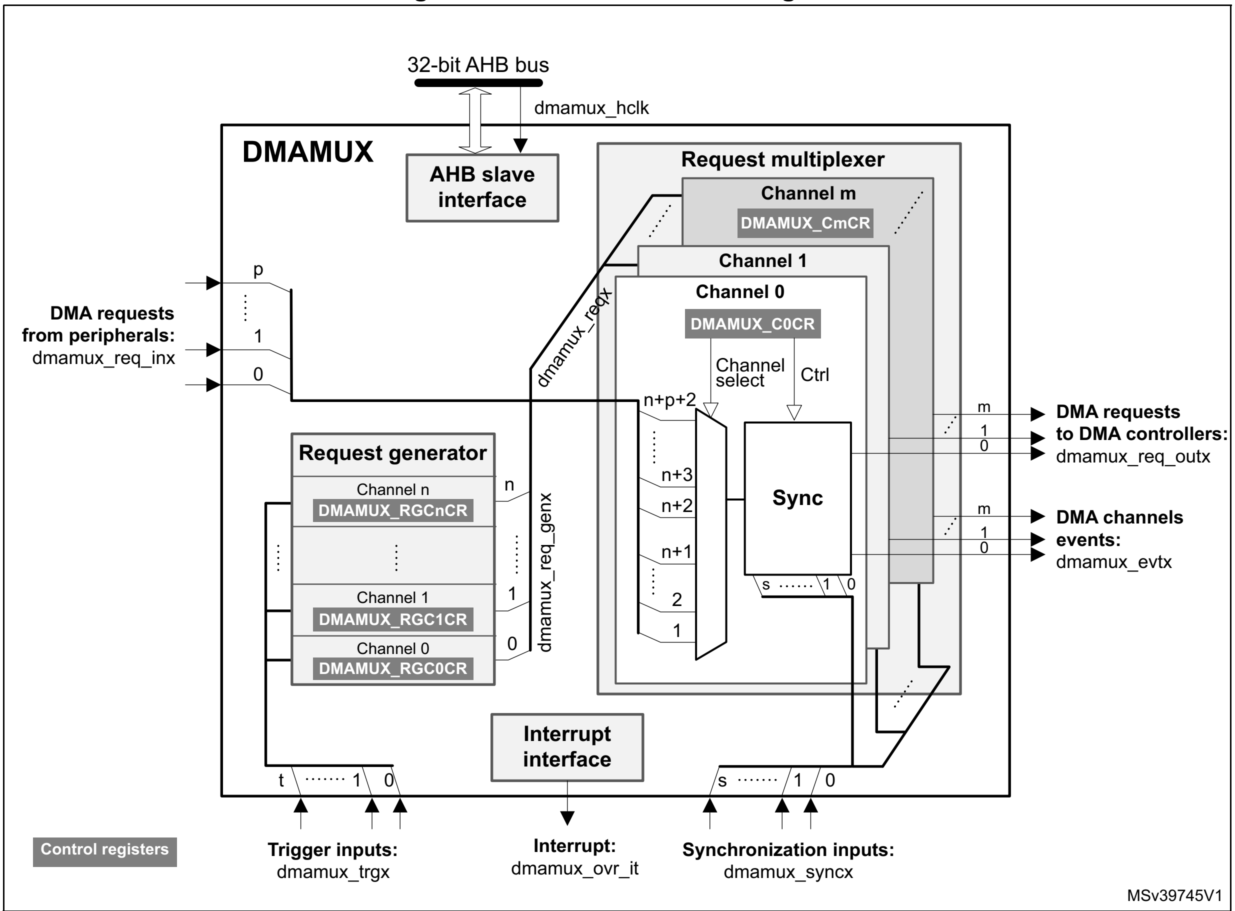
Sync. input	Resource	Sync. input	Resource
0	dmamux_evt0	5	LPTIMER2_OUT
1	dmamux_evt1	6	LPTIMER3_OUT
2	dmamux_evt2	7	extit0
3	LPTIMER1_OUT	8	TIM12_TRGO

19.4 DMAMUX functional description

19.4.1 DMAMUX block diagram

Figure 133 shows the DMAMUX block diagram.

Figure 133. DMAMUX block diagram



DMAMUX features two main sub-blocks: the request line multiplexer and the request line generator.

The implementation assigns:

- DMAMUX request multiplexer sub-block inputs (dmamux_reqx) from peripherals (dmamux_req_inx) and from channels of the DMAMUX request generator sub-block (dmamux_req_genx)
- DMAMUX request outputs to channels of DMA controllers (dmamux_req_outx)
- Internal or external signals to DMA request trigger inputs (dmamux_trgx)
- Internal or external signals to synchronization inputs (dmamux_syncx)

19.4.2 DMAMUX signals

Table 114 lists the DMAMUX signals.

Table 114. DMAMUX signals

Signal name	Description
dmamux_hclk	DMAMUX AHB clock
dmamux_req_inx	DMAMUX DMA request line inputs from peripherals
dmamux_trgx	DMAMUX DMA request triggers inputs (to request generator sub-block)
dmamux_req_genx	DMAMUX request generator sub-block channels outputs
dmamux_reqx	DMAMUX request multiplexer sub-block inputs (from peripheral requests and request generator channels)
dmamux_syncx	DMAMUX synchronization inputs (to request multiplexer sub-block)
dmamux_req_outx	DMAMUX requests outputs (to DMA controllers)
dmamux_evtx	DMAMUX events outputs
dmamux_ovr_it	DMAMUX overrun interrupts

19.4.3 DMAMUX channels

A DMAMUX channel is a request multiplexer channel that can include, depending upon the selected input of the request multiplexer, an additional DMAMUX request generator channel.

A DMAMUX request multiplexer channel is connected and dedicated to a single channel of DMA controller(s).

Channel configuration procedure

Follow the sequence below to configure a DMAMUX x channel and the related DMA channel y:

1. Set and configure completely the DMA channel y, except enabling the channel y.
2. Set and configure completely the related DMAMUX y channel.
3. Last, activate the DMA channel y by setting the EN bit in the DMA y channel register.

19.4.4 DMAMUX request line multiplexer

The DMAMUX request multiplexer with its multiple channels ensures the actual routing of DMA request/acknowledge control signals, named DMA request lines.

Each DMA request line is connected in parallel to all the channels of the DMAMUX request line multiplexer.

A DMA request is sourced either from the peripherals, or from the DMAMUX request generator.

The DMAMUX request line multiplexer channel x selects the DMA request line number as configured by the DMAREQ_ID field in the DMAMUX_CxCR register.

Note: The null value in the field DMAREQ_ID corresponds to no DMA request line selected.

Caution: A same non-null DMAREQ_ID cannot be programmed to different x and y DMAMUX request multiplexer channels (via DMAMUX_CxCR and DMAMUX_CyCR), except when the application guarantees that the two connected DMA channels are not simultaneously active.

On top of the DMA request selection, the synchronization mode and/or the event generation may be configured and enabled, if required.

Synchronization mode and channel event generation

Each DMAMUX request line multiplexer channel x can be individually synchronized by setting the synchronization enable (SE) bit in the DMAMUX_CxCR register.

DMAMUX has multiple synchronization inputs. The synchronization inputs are connected in parallel to all the channels of the request multiplexer.

The synchronization input is selected via the SYNC_ID field in the DMAMUX_CxCR register of a given channel x.

When a channel is in this synchronization mode, the selected input DMA request line is propagated to the multiplexer channel output, once a programmable rising/falling edge is detected on the selected input synchronization signal, via the SPOL[1:0] field of the DMAMUX_CxCR register.

Additionally, internally to the DMAMUX request multiplexer, there is a programmable DMA request counter, which can be used for the channel request output generation, and for an event generation. An event generation on the channel x output is enabled through the EGE bit (event generation enable) of the DMAMUX_CxCR register.

As shown in *Figure 135*, upon the detected edge of the synchronization input, the pending selected input DMA request line is connected to the DMAMUX multiplexer channel x output.

Note: If a synchronization event occurs while there is no pending selected input DMA request line, it is discarded. The following asserted input request lines is not connected to the DMAMUX multiplexer channel output until a synchronization event occurs again.

From this point on, each time the connected DMAMUX request is served by the DMA controller (a served request is deasserted), the DMAMUX request counter is decremented. At its underrun, the DMA request counter is automatically loaded with the value in the NBREQ field of the DMAMUX_CxCR register and the input DMA request line is disconnected from the multiplexer channel x output.

Thus, the number of DMA requests transferred to the multiplexer channel x output following a detected synchronization event, is equal to the value in the NBREQ field, plus one.

Note: The NBREQ field value can be written by software only when both synchronization enable bit (SE) and event generation enable bit (EGE) of the corresponding multiplexer channel x are disabled.

Figure 134. Synchronization mode of the DMAMUX request line multiplexer channel

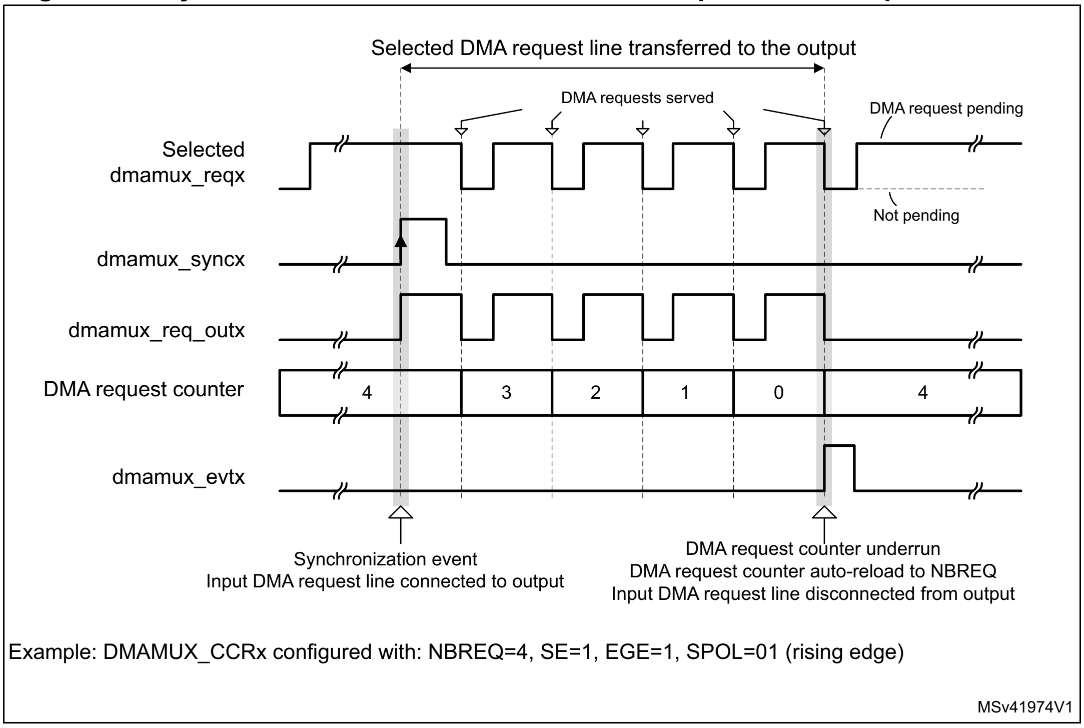
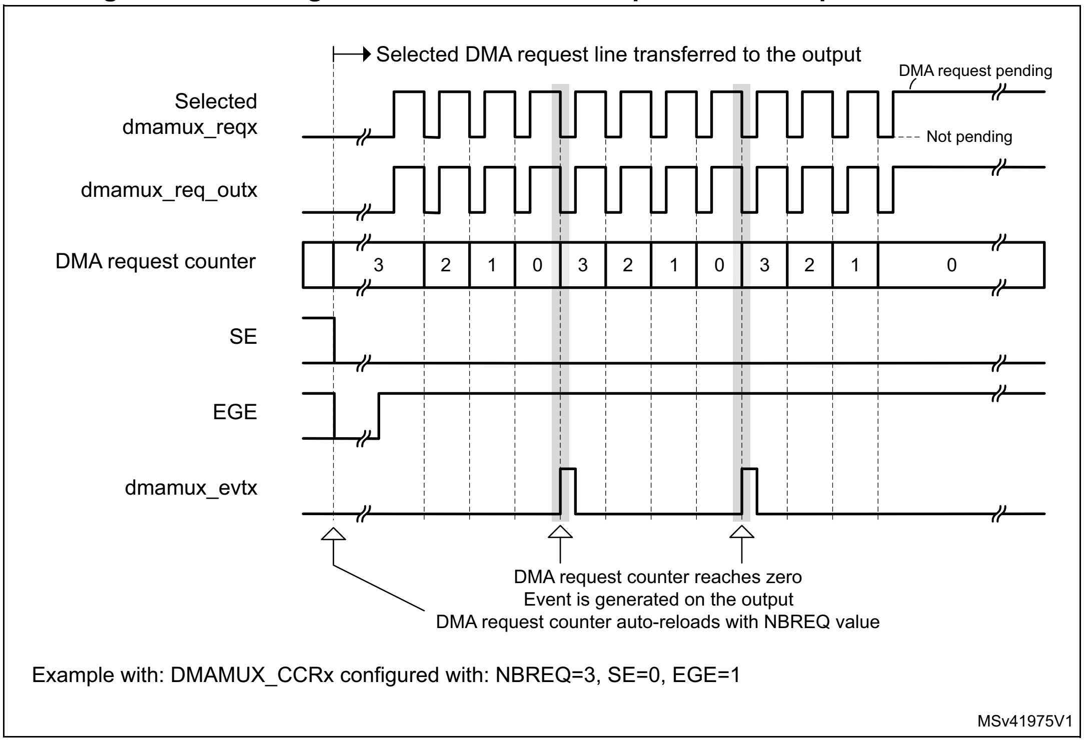


Figure 135. Event generation of the DMA request line multiplexer channel



If EGE is enabled, the multiplexer channel generates a channel event, as a pulse of one AHB clock cycle, when its DMA request counter is automatically reloaded with the value of the programmed NBREQ field, as shown in *Figure 134* and *Figure 135*.

Note: If EGE is enabled and NBREQ = 0, an event is generated after each served DMA request.

Note: A synchronization event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_CxCR register, the synchronization events are masked during three AHB clock cycles.

Synchronization overrun and interrupt

If a new synchronization event occurs before the request counter underrun (the internal request counter programmed via the NBREQ field of the DMAMUX_CxCR register), the synchronization overrun flag bit SOFx is set in the DMAMUX_CSR register.

Note: The request multiplexer channel x synchronization must be disabled (DMAMUX_CxCR.SE = 0) when the use of the related channel of the DMA controller is completed. Else, upon a new detected synchronization event, there is a synchronization overrun due to the absence of a DMA acknowledge (that is, no served request) received from the DMA controller.

The overrun flag SOFx is reset by setting the associated clear synchronization overrun flag bit CSOFx in the DMAMUX_CFR register.

Setting the synchronization overrun flag generates an interrupt if the synchronization overrun interrupt enable bit SOIE is set in the DMAMUX_CxCR register.

19.4.5 DMAMUX request generator

The DMAMUX request generator produces DMA requests following trigger events on its DMA request trigger inputs.

The DMAMUX request generator has multiple channels. DMA request trigger inputs are connected in parallel to all channels.

The outputs of DMAMUX request generator channels are inputs to the DMAMUX request line multiplexer.

Each DMAMUX request generator channel x has an enable bit GE (generator enable) in the corresponding DMAMUX_RGxCR register.

The DMA request trigger input for the DMAMUX request generator channel x is selected through the SIG_ID (trigger signal ID) field in the corresponding DMAMUX_RGxCR register.

Trigger events on a DMA request trigger input can be rising edge, falling edge or either edge. The active edge is selected through the GPOL (generator polarity) field in the corresponding DMAMUX_RGxCR register.

Upon the trigger event, the corresponding generator channel starts generating DMA requests on its output. Each time the DMAMUX generated request is served by the connected DMA controller (a served request is deasserted), a built-in (inside the DMAMUX request generator) DMA request counter is decremented. At its underrun, the request generator channel stops generating DMA requests and the DMA request counter is automatically reloaded to its programmed value upon the next trigger event.

Thus, the number of DMA requests generated after the trigger event is GNBREQ + 1.

Note: The GNBREQ field value can be written by software only when the enable GE bit of the corresponding generator channel x is disabled.

There is no hardware write protection.

A trigger event (edge) is detected if the state following the edge remains stable for more than two AHB clock cycles.

Upon writing into DMAMUX_RGxCR register, the trigger events are masked during three AHB clock cycles.

Trigger overrun and interrupt

If a new DMA request trigger event occurs before the DMAMUX request generator counter underrun (the internal counter programmed via the GNBREQ field of the DMAMUX_RGxCR register), and if the request generator channel x was enabled via GE, then the request trigger event overrun flag bit OFx is asserted by the hardware in the DMAMUX_RGSR register.

Note: The request generator channel x must be disabled (DMAMUX_RGxCR.GE = 0) when the usage of the related channel of the DMA controller is completed. Else, upon a new detected trigger event, there is a trigger overrun due to the absence of an acknowledge (that is, no served request) received from the DMA.

The overrun flag OFx is reset by setting the associated clear overrun flag bit COFx in the DMAMUX_RGCFR register.

Setting the DMAMUX request trigger overrun flag generates an interrupt if the DMA request trigger event overrun interrupt enable bit OIE is set in the DMAMUX_RGxCR register.

19.5 DMAMUX interrupts

An interrupt can be generated upon:

- a synchronization event overrun in each DMA request line multiplexer channel
- a trigger event overrun in each DMA request generator channel

For each case, per-channel in.dividual interrupt enable, status, and clear flag register bits are available.

Table 115. DMAMUX interrupts

Interrupt signal	Interrupt event	Event flag	Clear bit	Enable bit
dmamuxovr_it	Synchronization event overrun on channel x of the DMAMUX request line multiplexer	SOFx	CSOFx	SOIE
	Trigger event overrun on channel x of the DMAMUX request generator	OFx	COFx	OIE

19.6 DMAMUX registers

Refer to the table containing register boundary addresses for the DMAMUX base address.

DMAMUX registers may be accessed per byte (8-bit), half-word (16-bit), or word (32-bit). The address must be aligned with the data size.

19.6.1 DMAMUX request line multiplexer channel x configuration register (DMAMUX_CxCR)

Address offset: 0x000 + 0x04 * x (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
						rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **SYNC_ID[2:0]**: Synchronization identification
Selects the synchronization input (see *Table 113: DMAMUX: assignment of synchronization inputs to resources*).

Bits 23:19 **NBREQ[4:0]**: Number of DMA requests minus 1 to forward
Defines the number of DMA requests to forward to the DMA controller after a synchronization event, and/or the number of DMA requests before an output event is generated.
This field must only be written when both SE and EGE bits are low.

Bits 18:17 **SPOL[1:0]**: Synchronization polarity
Defines the edge polarity of the selected synchronization input:
00: No event (no synchronization, no detection).
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 16 **SE**: Synchronization enable
0: Synchronization disabled
1: Synchronization enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **EGE**: Event generation enable
0: Event generation disabled
1: Event generation enabled

Bit 8 **SOIE**: Synchronization overrun interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **DMAREQ_ID[6:0]**: DMA request identification
Selects the input DMA request. See the DMAMUX table about assignments of multiplexer inputs to resources.

19.6.2 DMAMUX request line multiplexer interrupt channel status register (DMAMUX_CSR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SOF15	SOF14	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SOF[15:0]**: Synchronization overrun event flag
The flag is set when a synchronization event occurs on a DMA request line multiplexer channel x, while the DMA request counter value is lower than NBREQ.
The flag is cleared by writing 1 to the corresponding CSOFx bit in DMAMUX_CFR register.

19.6.3 DMAMUX request line multiplexer interrupt clear flag register (DMAMUX_CFR)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CSOF 15	CSOF 14	CSOF 13	CSOF 12	CSOF 11	CSOF 10	CSOF 9	CSOF 8	CSOF 7	CSOF 6	CSOF 5	CSOF 4	CSOF 3	CSOF 2	CSOF 1	CSOF 0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSOF[15:0]**: Clear synchronization overrun event flag
Writing 1 in each bit clears the corresponding overrun flag SOFx in the DMAMUX_CSR register.

19.6.4 DMAMUX request generator channel x configuration register (DMAMUX_RGxCR)

Address offset: 0x100 + 0x04 * x (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL[1:0]		GE
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID[2:0]		
							rw						rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:19 **GNBREQ[4:0]**: Number of DMA requests to be generated (minus 1)

Defines the number of DMA requests to be generated after a trigger event. The actual number of generated DMA requests is GNBREQ +1.

Note: This field must be written only when GE bit is disabled.

Bits 18:17 **GPOL[1:0]**: DMA request generator trigger polarity

Defines the edge polarity of the selected trigger input

00: No event, i.e. no trigger detection nor generation.

01: Rising edge

10: Falling edge

11: Rising and falling edges

Bit 16 **GE**: DMA request generator channel x enable

0: DMA request generator channel x disabled

1: DMA request generator channel x enabled

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **OIE**: Trigger overrun interrupt enable

0: Interrupt on a trigger overrun event occurrence is disabled

1: Interrupt on a trigger overrun event occurrence is enabled

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **SIG_ID[2:0]**: Signal identification

Selects the DMA request trigger input used for the channel x of the DMA request generator

19.6.5 DMAMUX request generator interrupt status register (DMAMUX_RGSR)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF7	OF6	OF5	OF4	OF3	OF2	OF1	OF0
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **OF[7:0]**: Trigger overrun event flag

The flag is set when a new trigger event occurs on DMA request generator channel x, before the request counter underrun (the internal request counter programmed via the GNBREQ field of the DMAMUX_RGxCR register).

The flag is cleared by writing 1 to the corresponding COFx bit in the DMAMUX_RGCFR register.

19.6.6 DMAMUX request generator interrupt clear flag register (DMAMUX_RGCFR)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF7	COF6	COF5	COF4	COF3	COF2	COF1	COF0
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **COF[7:0]**: Clear trigger overrun event flag

Writing 1 in each bit clears the corresponding overrun flag OFx in the DMAMUX_RGSR register.

19.6.7 DMAMUX size identification register (DMAMUX_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification
This register identifies an IP size of 1 KB address space.

19.6.8 DMAMUX IP identification register (DMAMUX_IPIDR)

Address offset: 0x3F8

Reset value: 0x0010 0011

This register identifies the IP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** IP identification

19.6.9 DMAMUX version register (DMAMUX_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0011

This register identifies the IP version.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major IP revision

Bits 3:0 **MINREV[3:0]:** Minor IP revision

19.6.10 DMAMUX hardware configuration 1 register (DMAMUX_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x0808 6C10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUM_DMA_REQGEN[7:0]								NUM_DMA_TRIG[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NUM_DMA_PERIPH_REQ[7:0]								NUM_DMA_STREAMS[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **NUM_DMA_REQGEN[7:0]:** number of DMA request generator channels

Bits 23:16 **NUM_DMA_TRIG[7:0]:** number of synchronization inputs

Bits 15:8 **NUM_DMA_PERIPH_REQ[7:0]:** number of DMA request lines from peripherals

Bits 7:0 **NUM_DMA_STREAMS[7:0]:** number of DMA request line multiplexer (output) channels

19.6.11 DMAMUX hardware configuration 2 register (DMAMUX_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_DMA_EXT_REQ[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **NUM_DMA_EXT_REQ[7:0]:** Number of DMA request trigger inputs

19.6.12 DMAMUX register map

The following table summarizes the DMAMUX registers and reset values. Refer to the register boundary address table for the DMAMUX register base address.

Table 116. DMAMUX register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMAMUX_C0CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x004	DMAMUX_C1CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x008	DMAMUX_C2CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x00C	DMAMUX_C3CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x010	DMAMUX_C4CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x014	DMAMUX_C5CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x018	DMAMUX_C6CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x01C	DMAMUX_C7CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x020	DMAMUX_C8CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x024	DMAMUX_C9CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x028	DMAMUX_C10CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x02C	DMAMUX_C11CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x030	DMAMUX_C12CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x034	DMAMUX_C13CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x038	DMAMUX_C14CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value						0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x03C	DMAMUX_C15CR	Res.	Res.	Res.	Res.	Res.	SYNC_ID[2:0]			NBREQ[4:0]					SPOL[1:0]		SE	Res.	Res.	Res.	Res.	Res.	Res.	EGE	SOIE	Res.	DMAREQ_ID[6:0]						
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0							0	0		0	0	0	0	0	0	0
0x040 - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 116. DMAMUX register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	DMAMUX_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOF15	SOF14	SOF13	SOF12	SOF11	SOF10	SOF9	SOF8	SOF7	SOF6	SOF5	SOF4	SOF3	SOF2	SOF1	SOF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DMAMUX_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSOF15	CSOF14	CSOF13	CSOF12	CSOF11	CSOF10	CSOF9	CSOF8	CSOF7	CSOF6	CSOF5	CSOF4	CSOF3	CSOF2	CSOF1	CSOF0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088 - 0x0FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	DMAMUX_RG0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x104	DMAMUX_RG1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x108	DMAMUX_RG2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x10C	DMAMUX_RG3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x110	DMAMUX_RG4CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x114	DMAMUX_RG5CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x118	DMAMUX_RG6CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x11C	DMAMUX_RG7CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GNBREQ[4:0]					GPOL [1:0]		GE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIE	Res.	Res.	Res.	Res.	Res.	SIG_ID [3:0]		
	Reset value									0	0	0	0	0	0	0	0								0						0	0	0
0x120 - 0x13C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x140	DMAMUX_RGSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OF7	OF6	OF5	OF4	OF3	OF2	OF1	OF0
	Reset value																									0	0	0	0	0	0	0	0
0x144	DMAMUX_RGCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF7	COF6	COF5	COF4	COF3	COF2	COF1	COF0
	Reset value																									0	0	0	0	0	0	0	0
0x148 - 0x3E8	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x3EC	DMAMUX _HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_DMA_EXT_REQ							
	Reset value																									0	0	0	0	1	0	0	0
0x3F0	DMAMUX _HW_CFGR1	NUM_DMA_REQGEN								NUM_DMA_TRIG								NUM_DMA_PERIF_REQ								NUM_DMA_STREAMS							
	Reset value	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0
0x3F4	DMAMUX_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0x3F8	DMAMUX_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x3FC	DMAMUX_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

20 Interrupt list

20.1 NVIC interrupts

Table 117. STM32MP153 interrupt mapping for Cortex®-M4

Position	Priority	Type of priority	Acronym	Description	Address	EXTI event (1)
-	-	-	-	Reserved	0x0000 0000	-
-	-3	Fixed	Reset	Reset	0x0000 0004	-
-	-2	Fixed	NMI	Reserved	0x0000 0008	-
-	-1	Fixed	HardFault	All class of fault	0x0000 000C	-
-	0	Fixed	MemManage	Memory management	0x0000 0010	-
-	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014	-
-	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018	-
-	-	-	-	Reserved	0x0000 001C - 0x0000 002B	-
-	3	Settable	SVCall	System service call via SWI instruction	0x0000 002C	-
-	4	Settable	Debug Monitor	Debug monitor	0x0000 0030	-
-	-	-	-	Reserved	0x0000 0034	-
-	5	Settable	PendSV	Pendable request for system service	0x0000 0038	-
-	6	Settable	Systick	System tick timer	0x0000 003C	-
0	7	Settable	WWDG1_IT	Window watchdog 1 early wakeup interrupt	0x00000040	-
1	8	Settable	PVD_AVD	PVD & AVD detector through EXTI	0x00000044	16
2	9	Settable	TAMP	Tamper interrupt (include LSECSS interrupts)	0x00000048	(18)
3	10	Settable	RTC_WKUP_ALARM	RTC wakeup timer and alarms (A and B) interrupt	0x0000004C	(19)
4	11	Settable	-	-	0x00000050	-
5	12	Settable	RCC	RCC global interrupt (rcc_mcu_irq)	0x00000054	-
6	13	Settable	EXTI0	EXTI line 0 interrupt	0x00000058	0
7	14	Settable	EXTI1	EXTI line 1 interrupt	0x0000005C	1
8	15	Settable	EXTI2	EXTI line 2 interrupt	0x00000060	2
9	16	Settable	EXTI3	EXTI line 3 interrupt	0x00000064	3
10	17	Settable	EXTI4	EXTI line 4 interrupt	0x00000068	4
11	18	Settable	DMA1_STR0	DMA1 stream0 global interrupt	0x0000006C	-
12	19	Settable	DMA1_STR1	DMA1 stream1 global interrupt	0x00000070	-
13	20	Settable	DMA1_STR2	DMA1 stream2 global interrupt	0x00000074	-

Table 117. STM32MP153 interrupt mapping for Cortex®-M4 (continued)

Position	Priority	Type of priority	Acronym	Description	Address	EXTI event (1)
14	21	Settable	DMA1_STR3	DMA1 stream3 global interrupt	0x00000078	-
15	22	Settable	DMA1_STR4	DMA1 stream4 global interrupt	0x0000007C	-
16	23	Settable	DMA1_STR5	DMA1 stream5 global interrupt	0x00000080	-
17	24	Settable	DMA1_STR6	DMA1 stream6 global interrupt	0x00000084	-
18	25	Settable	ADC1	ADC1 global interrupt	0x00000088	-
19	26	Settable	FDCAN1_IT0	FDCAN1 interrupt 0	0x0000008C	-
20	27	Settable	FDCAN2_IT0	FDCAN2 interrupt 0	0x00000090	-
21	28	Settable	FDCAN1_IT1	FDCAN1 interrupt 1	0x00000094	-
22	29	Settable	FDCAN2_IT1	FDCAN2 interrupt 1	0x00000098	-
23	30	Settable	EXTI5	EXTI line 5 interrupt	0x0000009C	5
24	31	Settable	TIM1_BRK	TIM1 break interrupt	0x000000A0	-
25	32	Settable	TIM1_UP	TIM1 update interrupt	0x000000A4	-
26	33	Settable	TIM1_TRG_COM	TIM1 trigger and commutation interrupts	0x000000A8	-
27	34	Settable	TIM1_CC	TIM1 capture compare interrupt	0x000000AC	-
28	35	Settable	TIM2	TIM2 global interrupt	0x000000B0	-
29	36	Settable	TIM3	TIM3 global interrupt	0x000000B4	-
30	37	Settable	TIM4	TIM4 global interrupt	0x000000B8	-
31	38	Settable	I2C1_EVT	I2C1 event interrupt	0x000000BC	(21)
32	39	Settable	I2C1_ERR	I2C1 global error interrupt	0x000000C0	-
33	40	Settable	I2C2_EVT	I2C2 event interrupt	0x000000C4	(22)
34	41	Settable	I2C2_ERR	I2C2 global error interrupt	0x000000C8	-
35	42	Settable	SPI1	SPI1 global interrupt	0x000000CC	(36)
36	43	Settable	SPI2	SPI2 global interrupt	0x000000D0	(37)
37	44	Settable	USART1	USART1 global interrupt	0x000000D4	(26)
38	45	Settable	USART2	USART2 global interrupt	0x000000D8	(27)
39	46	Settable	USART3	USART3 global interrupt	0x000000DC	(28)
40	47	Settable	EXTI10	EXTI line 10 interrupt	0x000000E0	10
41	48	Settable	RTC_TS	RTC timestamp interrupt	0x000000E4	(17)
42	49	Settable	EXTI11	EXTI line 11 interrupt	0x000000E8	11
43	50	Settable	TIM8_BRK	TIM8 break interrupt	0x000000EC	-
44	51	Settable	TIM8_UP	TIM8 update interrupt	0x000000F0	-
45	52	Settable	TIM8_TRG_COM	TIM8 trigger & commutation interrupt	0x000000F4	-
46	53	Settable	TIM8_CC	TIM8 capture compare interrupt	0x000000F8	-

Table 117. STM32MP153 interrupt mapping for Cortex®-M4 (continued)

Position	Priority	Type of priority	Acronym	Description	Address	EXTI event (1)
47	54	Settable	DMA1_STR7	DMA1 stream7 global interrupt	0x000000FC	-
48	55	Settable	FMC	FMC global interrupt	0x00000100	-
49	56	Settable	SDMMC1	SDMMC global interrupt	0x00000104	-
50	57	Settable	TIM5	TIM5 global interrupt	0x00000108	-
51	58	Settable	SPI3	SPI3 global interrupt	0x0000010C	(38)
52	59	Settable	UART4	UART4 global interrupt	0x00000110	(30)
53	60	Settable	UART5	UART5 global interrupt	0x00000114	(31)
54	61	Settable	TIM6	TIM6 global interrupt	0x00000118	-
55	62	Settable	TIM7	TIM7 global interrupt	0x0000011C	-
56	63	Settable	DMA2_STR0	DMA2 stream0 global interrupt	0x00000120	-
57	64	Settable	DMA2_STR1	DMA2 stream1 global interrupt	0x00000124	-
58	65	Settable	DMA2_STR2	DMA2 stream2 global interrupt	0x00000128	-
59	66	Settable	DMA2_STR3	DMA2 stream3 global interrupt	0x0000012C	-
60	67	Settable	DMA2_STR4	DMA2 stream4 global interrupt	0x00000130	-
61	68	Settable	ETH1	ETH1 global interrupt	0x00000134	-
62	69	Settable	ETH1_WKUP	ETH1 wakeup interrupt (PMT)	0x00000138	(70)
63	70	Settable	FDCAN_CAL	FDCAN CCU interrupt	0x0000013C	-
64	71	Settable	EXTI6	EXTI line 6 interrupt	0x00000140	6
65	72	Settable	EXTI7	EXTI line 7 interrupt	0x00000144	7
66	73	Settable	EXTI8	EXTI line 8 interrupt	0x00000148	8
67	74	Settable	EXTI9	EXTI line 9 interrupt	0x0000014C	9
68	75	Settable	DMA2_STR5	DMA2 stream5 global interrupt	0x00000150	-
69	76	Settable	DMA2_STR6	DMA2 stream6 global interrupt	0x00000154	-
70	77	Settable	DMA2_STR7	DMA2 stream7 global interrupt	0x00000158	-
71	78	Settable	USART6	USART6 global interrupt	0x0000015C	(29)
72	79	Settable	I2C3_EVT	I2C3 event interrupt	0x00000160	(23)
73	80	Settable	I2C3_ERR	I2C3 error interrupt	0x00000164	-
74	81	Settable	USBH_OHCI	USB host OHCI interrupt	0x00000168	(43)
75	82	Settable	USBH_EHCI	USB host EHCI Interrupt	0x0000016C	(43)
76	83	Settable	EXTI12	EXTI line 12 interrupt	0x00000170	12
77	84	Settable	EXTI13	EXTI line 13 interrupt	0x00000174	13
78	85	Settable	DCMI	DCMI global interrupt	0x00000178	-
79	86	Settable	CRYP1(2)	CRYP1 global interrupt	0x0000017C	-

Table 117. STM32MP153 interrupt mapping for Cortex®-M4 (continued)

Position	Priority	Type of priority	Acronym	Description	Address	EXTI event (1)
80	87	Settable	HASH1	HASH1 interrupt	0x00000180	-
81	88	Settable	FPU	FPU global interrupt	0x00000184	-
82	89	Settable	UART7	UART7 global interrupt	0x00000188	(32)
83	90	Settable	UART8	UART8 global interrupt	0x0000018C	(33)
84	91	Settable	SPI4	SPI4 global interrupt	0x00000190	(39)
85	92	Settable	SPI5	SPI5 global interrupt	0x00000194	(40)
86	93	Settable	SPI6	SPI6 global interrupt	0x00000198	(41)
87	94	Settable	SAI1	SAI1 global interrupt	0x0000019C	-
88	95	Settable	LTDC	LTCD global interrupt	0x000001A0	-
89	96	Settable	LTDC_ER	LTCD global error interrupt	0x000001A4	-
90	97	Settable	ADC2	ADC2 global interrupt	0x000001A8	-
91	98	Settable	SAI2	SAI2 global interrupt	0x000001AC	-
92	99	Settable	QUADSPI	QUADSPI global interrupt	0x000001B0	-
93	100	Settable	LPTIM1	LPTIMER1 global interrupt	0x000001B4	(47)
94	101	Settable	CEC	HDMI-CEC global interrupt	0x000001B8	(69)
95	102	Settable	I2C4_EVT	I2C4 event interrupt	0x000001BC	(24)
96	103	Settable	I2C4_ERR	I2C4 error interrupt	0x000001C0	-
97	104	Settable	SPDIFRX	SPDIFRX global interrupt	0x000001C4	-
98	105	Settable	OTG	USB On-The-Go global interrupt	0x000001C8	(44)
99	106	Settable	-	-	0x000001CC	-
100	107	Settable	IPCC_RX0	IPCC RX0 occupied interrupt	0x000001D0	(61)
101	108	Settable	IPCC_TX0	IPCC TX0 free interrupt	0x000001D4	(61)
102	109	Settable	DMAMUX1_OVR_REQ	DMAMUX1 overrun interrupt	0x000001D8	-
103	110	Settable	IPCC_RX1	IPCC RX1 occupied interrupt	0x000001DC	(62)
104	111	Settable	IPCC_TX1	IPCC TX1 free interrupt	0x000001E0	(62)
105	112	Settable	CRYP2(2)	CRYP2 global interrupt	0x000001E4	-
106	113	Settable	HASH2	HASH2 interrupt	0x000001E8	-
107	114	Settable	I2C5_EVT	I2C5 event interrupt	0x000001EC	(25)
108	115	Settable	I2C5_ERR	I2C5 error interrupt	0x000001F0	-
109	116	-	-	Reserved	0x000001F4	-
110	117	Settable	DFSDM1_FLT0	DFSDM1 filter0 Interrupt	0x000001F8	-
111	118	Settable	DFSDM1_FLT1	DFSDM1 filter1 Interrupt	0x000001FC	-
112	119	Settable	DFSDM1_FLT2	DFSDM1 filter2 Interrupt	0x00000200	-

Table 117. STM32MP153 interrupt mapping for Cortex®-M4 (continued)

Position	Priority	Type of priority	Acronym	Description	Address	EXTI event (1)
113	120	Settable	DFSDM1_FLT3	DFSDM1 filter3 Interrupt	0x00000204	-
114	121	Settable	SAI3	SAI3 global interrupt	0x00000208	-
115	122	Settable	DFSDM1_FLT4	DFSDM1 filter4 Interrupt	0x0000020C	-
116	123	Settable	TIM15	TIM15 global interrupt	0x00000210	-
117	124	Settable	TIM16	TIM16 global interrupt	0x00000214	-
118	125	Settable	TIM17	TIM17 global interrupt	0x00000218	-
119	126	Settable	TIM12	TIM12 global interrupt	0x0000021C	-
120	127	Settable	MDIOS	MDIOS global interrupt	0x00000220	(42)
121	128	Settable	EXTI14	EXTI line 14 interrupt	0x00000224	14
122	129	Settable	MDMA	MDMA global interrupt	0x00000228	-
123	130	-	-	Reserved	0x0000022C	-
124	131	Settable	SDMMC2	SDMMC2 global interrupt	0x00000230	-
125	132	Settable	HSEM_IT2	HSEM semaphore interrupt 2	0x00000234	(64)
126	133	Settable	DFSDM1_FLT5	DFSDM1 filter5 interrupt	-	-
127	134	Settable	EXTI15	EXTI line 15 interrupt	0x0000023C	15
128	135	Settable	nCTIIRQ1	Cortex®-M4 CTI interrupt 1	0x00000240	-
129	136	Settable	nCTIIRQ2	Cortex®-M4 CTI interrupt 2	0x00000244	-
130	137	Settable	TIM13	TIM13 global interrupt	0x00000248	-
131	138	Settable	TIM14	TIM14 global interrupt	0x0000024C	-
132	139	Settable	DAC	DAC1 and DAC2 underrun error interrupts	0x00000250	-
133	140	Settable	RNG1	RNG1 interrupt	0x00000254	-
134	141	Settable	RNG2	RNG2 interrupt	0x00000258	-
135	142	Settable	I2C6_EVT	I2C6 event interrupt	0x0000025C	(54)
136	143	Settable	I2C6_ERR	I2C6 error interrupt	0x00000260	-
137	144	Settable	SDMMC3	SDMMC global interrupt	0x00000264	-
138	145	Settable	LPTIM2	LPTIMER2 global interrupt	0x00000268	(48)
139	146	Settable	LPTIM3	LPTIMER3 global interrupt	0x0000026C	(50)
140	147	Settable	LPTIM4	LPTIMER4 global interrupt	0x00000270	(52)
141	148	Settable	LPTIM5	LPTIMER5 global interrupt	0x00000274	(53)
142	149	Settable	ETH1_LPI	ETH1 LPI interrupt	0x00000278	(71)
143	150	Settable	-	-	0x0000027C	-
144	151	Settable	MPU_SEV	Cortex®-A7 send event through EXTI	0x00000280	66
145	152	Settable	RCC_WAKEUP	RCC MCU wakeup interrupt	0x00000284	-

Table 117. STM32MP153 interrupt mapping for Cortex®-M4 (continued)

Position	Priority	Type of priority	Acronym	Description	Address	EXTI event (1)
146	153	Settable	SAI4	SAI4 global interrupt	0x00000288	-
147	154	Settable	DTS	Digital temperature sensor interrupt	0x0000028C	(72)
148	155	Settable	-	-	0x00000290	-
149	156	Settable	MCU_WAKEUP_PIN	Interrupt for all 6 wakeup enabled by MCU	0x00000294	(55, 56, 57, 58, 59, 60)

1. - The absence of parenthesis around EXTI event numbers indicates that the EXTI output is connected to the NVIC. The EXTI configuration may impact the interrupt line state (i.e. EXTI may mask the interrupt).
   - The presence of parenthesis () around EXTI event numbers indicates that the EXTI output is not connected to the NVIC. The EXTI configuration cannot impact the interrupt line state.
   See *Section 23: Extended interrupt and event controller (EXTI)* for details.
2. Check device datasheet for availability.

20.2 GIC Interrupts

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC

Num	ID	Acronym	Description	EXTI event (1)
SGI (Rising Edge triggered)				
0	0	SGI0	software generated interrupt 0 (recommended non-secure)	-
1	1	SGI1	software generated interrupt 1 (recommended non-secure)	-
2	2	SGI2	software generated interrupt 2 (recommended non-secure)	-
3	3	SGI3	software generated interrupt 3 (recommended non-secure)	-
4	4	SGI4	software generated interrupt 4 (recommended non-secure)	-
5	5	SGI5	software generated interrupt 5 (recommended non-secure)	-
6	6	SGI6	software generated interrupt 6 (recommended non-secure)	-
7	7	SGI7	software generated interrupt 7 (recommended non-secure)	-
8	8	SGI8	software generated interrupt 8 (recommended secure)	-
9	9	SGI9	software generated interrupt 9 (recommended secure)	-
10	10	SGI10	software generated interrupt 10 (recommended secure)	-
11	11	SGI11	software generated interrupt 11 (recommended secure)	-
12	12	SGI12	software generated interrupt 12 (recommended secure)	-
13	13	SGI13	software generated interrupt 13 (recommended secure)	-
14	14	SGI14	software generated interrupt 14 (recommended secure)	-
15	15	SGI15	software generated interrupt 15 (recommended secure)	-

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC (continued)

Num	ID	Acronym	Description	EXTI event (1)
PPI (Active Low level sensitive)				
-	16 to 24	-	Reserved	-
-	25	PPI6	Virtual maintenance interrupt.	-
-	26	PPI5	Hypervisor timer event.	-
-	27	PPI4	Virtual timer event.	-
-	28	PPI0	Legacy nFIQ signal. Not used.	-
-	29	PPI1	Secure physical timer event.	-
-	30	PPI2	Non-secure physical timer event.	-
-	31	PPI3	Legacy nIRQ signal. Not used.	-
SPI (active high level sensitive or rising edge triggered)				
0	32	WWDG1_IT	Window watchdog 1 early wakeup interrupt	-
1	33	PVD_AVD	PVD & AVD detector through EXTI	16
2	34	TAMP	Tamper interrupt (include LSECSS interrupts)	(18)
3	35	RTC_WKUP_ALARM	RTC wakeup timer and alarms (A and B) interrupt	(19)
4	36	TZC_IT	TrustZone DDR address space controller	-
5	37	RCC	RCC global interrupt	-
6	38	EXTI0	EXTI line 0 interrupt	0
7	39	EXTI1	EXTI line 1 interrupt	1
8	40	EXTI2	EXTI line 2 interrupt	2
9	41	EXTI3	EXTI line 3 interrupt	3
10	42	EXTI4	EXTI line 4 interrupt	4
11	43	DMA1_STR0	DMA1 stream0 global interrupt	-
12	44	DMA1_STR1	DMA1 stream1 global interrupt	-
13	45	DMA1_STR2	DMA1 stream2 global interrupt	-
14	46	DMA1_STR3	DMA1 stream3 global interrupt	-
15	47	DMA1_STR4	DMA1 stream4 global interrupt	-
16	48	DMA1_STR5	DMA1 stream5 global interrupt	-
17	49	DMA1_STR6	DMA1 stream6 global interrupt	-
18	50	ADC1	ADC1 global interrupt	-
19	51	FDCAN1_IT0	FDCAN1 interrupt 0	-
20	52	FDCAN2_IT0	FDCAN2 interrupt 0	-
21	53	FDCAN1_IT1	FDCAN1 interrupt 1	-
22	54	FDCAN2_IT1	FDCAN2 interrupt 1	-

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC (continued)

Num	ID	Acronym	Description	EXTI event (1)
23	55	EXTI5	EXTI line 5 interrupt	5
24	56	TIM1_BRK	TIM1 break interrupt	-
25	57	TIM1_UP	TIM1 update interrupt	-
26	58	TIM1_TRG_COM	TIM1 trigger and commutation interrupts	-
27	59	TIM1_CC	TIM1 capture compare interrupt	-
28	60	TIM2	TIM2 global interrupt	-
29	61	TIM3	TIM3 global interrupt	-
30	62	TIM4	TIM4 global interrupt	-
31	63	I2C1_EVT	I2C1 event interrupt	(21)
32	64	I2C1_ERR	I2C1 global error interrupt	-
33	65	I2C2_EVT	I2C2 event interrupt	(22)
34	66	I2C2_ERR	I2C2 global error interrupt	-
35	67	SPI1	SPI1 global interrupt	(36)
36	68	SPI2	SPI2 global interrupt	(37)
37	69	USART1	USART1 global interrupt	(26)
38	70	USART2	USART2 global interrupt	(27)
39	71	USART3	USART3 global interrupt	(28)
40	72	EXTI10	EXTI line 10 interrupt	10
41	73	RTC_TS	RTC timestamp interrupt	(17)
42	74	EXTI11	EXTI line 11 interrupt	11
43	75	TIM8_BRK	TIM8 break interrupt	-
44	76	TIM8_UP	TIM8 update interrupt	-
45	77	TIM8_TRG_COM	TIM8 trigger & commutation interrupt	-
46	78	TIM8_CC	TIM8 capture compare interrupt	-
47	79	DMA1_STR7	DMA1 stream7 global interrupt	-
48	80	FMC	FMC global interrupt	-
49	81	SDMMC1	SDMMC global interrupt	-
50	82	TIM5	TIM5 global interrupt	-
51	83	SPI3	SPI3 global interrupt	(38)
52	84	UART4	UART4 global interrupt	(30)
53	85	UART5	UART5 global interrupt	(31)
54	86	TIM6	TIM6 global interrupt	-
55	87	TIM7	TIM7 global interrupt	-

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC (continued)

Num	ID	Acronym	Description	EXTI event (1)
56	88	DMA2_STR0	DMA2 stream0 global interrupt	-
57	89	DMA2_STR1	DMA2 stream1 global interrupt	-
58	90	DMA2_STR2	DMA2 stream2 global interrupt	-
59	91	DMA2_STR3	DMA2 stream3 global interrupt	-
60	92	DMA2_STR4	DMA2 stream4 global interrupt	-
61	93	ETH1	Ethernet global interrupt	-
62	94	ETH1_WKUP	ETHERNET wakeup interrupt (PMT)	(70)
63	95	FDCAN_CAL	FDCAN CCU interrupt	-
64	96	EXTI6	EXTI line 6 interrupt	6
65	97	EXTI7	EXTI line 7 interrupt	7
66	98	EXTI8	EXTI line 8 interrupt	8
67	99	EXTI9	EXTI line 9 interrupt	9
68	100	DMA2_STR5	DMA2 stream5 global interrupt	-
69	101	DMA2_STR6	DMA2 stream6 global interrupt	-
70	102	DMA2_STR7	DMA2 stream7 global interrupt	-
71	103	USART6	USART6 global interrupt	(29)
72	104	I2C3_EVT	I2C3 event interrupt	(23)
73	105	I2C3_ERR	I2C3 error interrupt	-
74	106	USBH_OHCI	USB host OHCI Interrupt	(43)
75	107	USBH_EHCI	USB host EHCI Interrupt	(43)
76	108	EXTI12	EXTI line 12 interrupt	12
77	109	EXTI13	EXTI line 13 interrupt	13
78	110	DCMI	DCMI global interrupt	-
79	111	CRYP1(2)	CRYP1 global interrupt	-
80	112	HASH1	HASH1 interrupt	-
81	113	-	Reserved	-
82	114	UART7	UART7 global interrupt	(32)
83	115	UART8	UART8 global interrupt	(33)
84	116	SPI4	SPI4 global interrupt	(39)
85	117	SPI5	SPI5 global interrupt	(40)
86	118	SPI6	SPI6 global interrupt	(41)
87	119	SAI1	SAI1 global interrupt	-
88	120	LTDC	LTCD global interrupt	-

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC (continued)

Num	ID	Acronym	Description	EXTI event (1)
89	121	LTDC_ER	LTCD global Error interrupt	-
90	122	ADC2	ADC2 global interrupt	-
91	123	SAI2	SAI2 global interrupt	-
92	124	QUADSPI	QUADSPI global interrupt	-
93	125	LPTIM1	LPTIMER1 global interrupt	(47)
94	126	CEC	HDMI-CEC global interrupt	(69)
95	127	I2C4_EVT	I2C4 event interrupt	(24)
96	128	I2C4_ERR	I2C4 error interrupt	-
97	129	SPDIFRX	SPDIFRX global interrupt	-
98	130	OTG	USB On-The-Go global interrupt	(44)
99	131	-	-	-
100	132	IPCC_RX0	IPCC RX0 occupied interrupt	(61)
101	133	IPCC_TX0	IPCC TX0 free interrupt	(61)
102	134	DMAMUX1_OVR_REQ	DMAMUX1 overrun interrupt	-
103	135	IPCC_RX1	IPCC RX1 Occupied interrupt	(62)
104	136	IPCC_TX1	IPCC TX1 Free interrupt	(62)
105	137	CRYP2(2)	CRYP2 global interrupt	-
106	138	HASH2	HASH2 interrupt	-
107	139	I2C5_EVT	I2C5 event interrupt	(25)
108	140	I2C5_ERR	I2C5 error interrupt	-
109	141	-	Reserved	-
110	142	DFSDM1_FLT0	DFSDM1 filter0 Interrupt	-
111	143	DFSDM1_FLT1	DFSDM1 filter1 Interrupt	-
112	144	DFSDM1_FLT2	DFSDM1 filter2 Interrupt	-
113	145	DFSDM1_FLT3	DFSDM1 filter3 Interrupt	-
114	146	SAI3	SAI3 global interrupt	-
115	147	DFSDM1_FLT4	DFSDM1 Filter4 Interrupt	-
116	148	TIM15	TIM15 global interrupt	-
117	149	TIM16	TIM16 global interrupt	-
118	150	TIM17	TIM17 global interrupt	-
119	151	TIM12	TIM12 global interrupt	-
120	152	MDIOS	MDIOS global interrupt	(42)
121	153	EXTI14	EXTI line 14 interrupt	14

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC (continued)

Num	ID	Acronym	Description	EXTI event (1)
122	154	MDMA	MDMA global interrupt	-
123	155	-	Reserved	-
124	156	SDMMC2	SDMMC2 global interrupt	-
125	157	HSEM_IT1	HSEM semaphore Interrupt 1	(63)
126	158	DFSDM1_FLT5	DFSDM1 filter5 Interrupt	-
127	159	EXTI15	EXTI line 15 interrupt	15
128	160	MDMA_SEC_IT	MDMA global secure interrupt	-
129	161	SYSRESETQ	MCU local reset request through EXTI	73
130	162	TIM13	TIM13 global interrupt	-
131	163	TIM14	TIM14 global interrupt	-
132	164	DAC	DAC1 and DAC2 underrun error interrupts	-
133	165	RNG1	RNG1 interrupt	-
134	166	RNG2	RNG2 interrupt	-
135	167	I2C6_EVT	I2C6 event interrupt	(54)
136	168	I2C6_ERR	I2C6 error interrupt	-
137	169	SDMMC3	SDMMC global interrupt	-
138	170	LPTIM2	LPTIMER2 global interrupt	(48)
139	171	LPTIM3	LPTIMER3 global interrupt	(50)
140	172	LPTIM4	LPTIMER4 global interrupt	(52)
141	173	LPTIM5	LPTIMER5 global interrupt	(53)
142	174	ETH1_LPI	ETH1 LPI interrupt	(71)
143	175	WWDG1_RST_IT	Window watchdog 1 reset through EXTI	68
144	176	MCU_SEV	Cortex®-M4 send event through EXTI	65
145	177	RCC_WAKEUP	RCC MPU wakeup interrupt	-
146	178	SAI4	SAI4 global interrupt	-
147	179	DTS	Digital temperature sensor interrupt	(72)
148	180	-	-	-
149	181	MPU_WAKEUP_PIN	Interrupt for all 6 wake-up enabled by MPU	(55, 56, 57, 58, 59, 60)
150	182	IWDG1_IT	IWDG1 early wake	(45)
151	183	IWDG2_IT	IWDG2 early wake	(46)
152 to 196	184 to 228	-	Reserved	-

Table 118. STM32MP153 interrupt mapping for Cortex®-A7 GIC (continued)

Num	ID	Acronym	Description	EXTI event (1)
197	229	TAMP_S	TAMP tamper secure interrupt	(18)
198	230	RTC_WKUP_ALARM_S	RTC wakeup timer and alarms (A and B) secure interrupt	(19)
199	231	RTC_TS_S	RTC timestamp secure interrupt	(17)
200	232	PMUIRQ0	Cortex®-A7 core#0 performance monitor interrupt	-
201	233	PMUIRQ1	Cortex®-A7 core#1 performance monitor interrupt	-
202	234	-	Reserved	-
203	235	-	Reserved	-
204	236	COMMRX0	Cortex®-A7 core#0 debug communication channel Receive interrupt	-
205	237	COMMRX1	Cortex®-A7 core#1 debug communication channel receive interrupt	-
206	238	-	Reserved	-
207	239	-	Reserved	-
208	240	COMMTX0	Cortex®-A7 core#0 debug communication channel transmit interrupt	-
209	241	COMMTX1	Cortex®-A7 core#1 debug communication channel transmit interrupt	-
210	242	-	Reserved	-
211	243	-	Reserved	-
212	244	AXIERRIRQ	Asynchronous AXI abort interrupt	-
213	245	DDRPERFM	DDR performance monitor interrupt	-
214	246	-	Reserved	-
215	247	-	Reserved	-
216	248	nCTIIRQ0	Cortex®-A7 core#0 CTI interrupt	-
217	249	nCTIIRQ1	Cortex®-A7 core#1 CTI interrupt	-
210	242	-	Reserved	-
211	243	-	Reserved	-
212 to 255	244 to 287	-	Reserved	-

1. - The absence of parenthesis around EXTI event numbers indicates that the EXTI output is connected to the GIC. The EXTI configuration may impact the interrupt line state (i.e. EXTI may mask the interrupt).
   - The presence of parenthesis () around EXTI event numbers indicates that the EXTI output is not connected to the GIC. The EXTI configuration cannot impact the interrupt line state.
   See *Section 23: Extended interrupt and event controller (EXTI)* for details.
2. Check device datasheet for availability.

20.3 EXTI events

Table 119. STM32MP153 EXTI Events

Event Input	Source	Event type	Wakeup target	Connection to rxev[(1)]	TrustZone
0	EXTI[0]	Configurable	MPU & MCU	Yes	Yes
1	EXTI[1]	Configurable	MPU & MCU	Yes	Yes
2	EXTI[2]	Configurable	MPU & MCU	Yes	Yes
3	EXTI[3]	Configurable	MPU & MCU	Yes	Yes
4	EXTI[4]	Configurable	MPU & MCU	Yes	Yes
5	EXTI[5]	Configurable	MPU & MCU	Yes	Yes
6	EXTI[6]	Configurable	MPU & MCU	Yes	Yes
7	EXTI[7]	Configurable	MPU & MCU	Yes	Yes
8	EXTI[8]	Configurable	MPU & MCU	Yes	Yes
9	EXTI[9]	Configurable	MPU & MCU	Yes	Yes
10	EXTI[10]	Configurable	MPU & MCU	Yes	Yes
11	EXTI[11]	Configurable	MPU & MCU	Yes	Yes
12	EXTI[12]	Configurable	MPU & MCU	Yes	Yes
13	EXTI[13]	Configurable	MPU & MCU	Yes	Yes
14	EXTI[14]	Configurable	MPU & MCU	Yes	Yes
15	EXTI[15]	Configurable	MPU & MCU	Yes	Yes
16	PVD and AVD	Configurable	MPU & MCU	No	No
17	RTC timestamp wakeup (ORed none-secure & secure)	Direct	MPU & MCU	Yes	Yes
18	TAMP tamper wakeup (ORed none-secure & secure)	Direct	MPU & MCU	Yes	Yes

Table 119. STM32MP153 EXTI Events (continued)

Event Input	Source	Event type	Wakeup target	Connection to rxev(1)	TrustZone
19	RTC wakeup timer and alarms (A and B) wakeup (ORed none-secure and secure)	Direct	MPU & MCU	Yes	Yes
20	Reserved	-	-	-	-
21	I2C1 wakeup	Direct	MPU & MCU	No	No
22	I2C2 wakeup	Direct	MPU & MCU	No	No
23	I2C3 wakeup	Direct	MPU & MCU	No	No
24	I2C4 wakeup	Direct	MPU & MCU	No	Yes
25	I2C5 wakeup	Direct	MPU & MCU	No	No
26	USART1 wakeup	Direct	MPU & MCU	No	Yes
27	USART2 wakeup	Direct	MPU & MCU	No	No
28	USART3 wakeup	Direct	MPU & MCU	No	No
29	USART6 wakeup	Direct	MPU & MCU	No	No
30	UART4 wakeup	Direct	MPU & MCU	No	No
31	UART5 wakeup	Direct	MPU & MCU	No	No
32	UART7 wakeup	Direct	MPU & MCU	No	No
33	UART8 wakeup	Direct	MPU & MCU	No	No
34	Reserved	-	-	-	-
35	Reserved	-	-	-	-
36	SPI1 wakeup	Direct	MPU & MCU	No	No
37	SPI2 wakeup	Direct	MPU & MCU	No	No
38	SPI3 wakeup	Direct	MPU & MCU	No	No
39	SPI4 wakeup	Direct	MPU & MCU	No	No

Table 119. STM32MP153 EXTI Events (continued)

Event Input	Source	Event type	Wakeup target	Connection to rxev(1)	TrustZone
40	SPI5 wakeup	Direct	MPU & MCU	No	No
41	SPI6 wakeup	Direct	MPU & MCU	No	Yes
42	MDIOS wakeup	Direct	MPU & MCU	No	No
43	USBH wakeup	Direct	MPU & MCU	No	No
44	OTG wakeup	Direct	MPU & MCU	No	No
45	IWDG1 early wake	Direct	MPU & MCU	No	Yes
46	IWDG2 early wake	Direct	MPU & MCU	No	No
47	LPTIM1 wakeup	Direct	MPU & MCU	No	No
48	LPTIM2 wakeup	Direct	MPU & MCU	No	No
49	Reserved	-	-	-	-
50	LPTIM3 wakeup	Direct	MPU & MCU	No	No
51	Reserved	-	-	-	-
52	LPTIM4 wakeup	Direct	MPU & MCU	No	No
53	LPTIM5 wakeup	Direct	MPU & MCU	No	No
54	I2C6 wakeup	Direct	MPU & MCU	No	Yes
55	WKUP1 wakeup	Direct	MPU & MCU	No	Yes
56	WKUP2 wakeup	Direct	MPU & MCU	No	Yes
57	WKUP3 wakeup	Direct	MPU & MCU	No	Yes
58	WKUP4 wakeup	Direct	MPU & MCU	No	Yes
59	WKUP5 wakeup	Direct	MPU & MCU	No	Yes

Table 119. STM32MP153 EXTI Events (continued)

Event Input	Source	Event type	Wakeup target	Connection to rxev[(1)]	TrustZone
60	WKUP6 wakeup	Direct	MPU & MCU	No	Yes
61	IPCC interrupt CPU1	Direct	MPU & MCU	No	No
62	IPCC interrupt CPU2	Direct	MPU & MCU	No	No
63	HSEM_IT1 interrupt	Direct	MPU only	No	No
64	HSEM_IT2 interrupt	Direct	MCU Only	No	No
65	CPU2 SEV interrupt	Configurable	MPU Only	No	No
66	CPU1 SEV interrupt	Configurable	MCU Only	Yes	No
67	Reserved	-	-	-	-
68	WWDG1 reset	Configurable	MPU only	No	No
69	HDMICEC wakeup	Direct	MPU & MCU	No	No
70	ETH1 pmt_intr_o wakeup	Direct	MPU & MCU	No	No
71	ETH1 lpi_intr_o wakeup	Direct	MPU & MCU	No	No
72	DTS wakeup	Direct	MPU & MCU	No	No
73	CPU2 SYSRESETREQ local CPU2 reset	Configurable	MPU only	No	No
74	Reserved	-	-	-	-
75	CDBGPWRUPREQ event	Direct	MPU & MCU	No	No

1. rxev is only present on MCU core.

21 Nested Vectored Interrupt Controllers (NVIC)

21.1 NVIC features

The nested vector interrupt controller NVIC includes the following features:

- 150 maskable interrupt channels
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the Cortex-M4 processor core are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All the interrupts including the Cortex-M4 core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming, refer to Cortex®-M4 programming manual PM0214.

21.2 SysTick calibration value register

As the Systick is not running on a fixed frequency, there is no SystTick calibration value available. The SysTick reload counter should be set according to the required time base and the mcu_ck frequency value set in RCC.

21.3 Interrupt and exception vectors

See the interrupt tables for STM32MP153 devices in *Interrupt list* section.

22 Global interrupt controller (GIC)

22.1 Introduction

The following Arm® specifications provides additional details on the GIC:

- Arm® Generic Interrupt Controller - Architecture Specification v2.0
- Cortex®-A7 MPCore Technical Reference Manual Revision: r0p5

22.2 GIC main features

The integrated GIC collates and arbitrates from a large number of interrupt sources.

It provides:

- Masking of interrupts
- Prioritization of interrupts
- Distribution of the interrupts to the target processors
- Tracking the status of interrupts
- Generation of interrupts by software
- Support for security extensions
- Support for virtualization extensions

22.3 GIC functional description

The GIC is a single functional unit located in the Cortex-A7 MPCore processor. The GIC consists of a shared distributor block and several CPU interfaces. For each processor in the multiprocessor device, there is:

- one CPU interface
- a virtual interface control
- a virtual CPU interface.

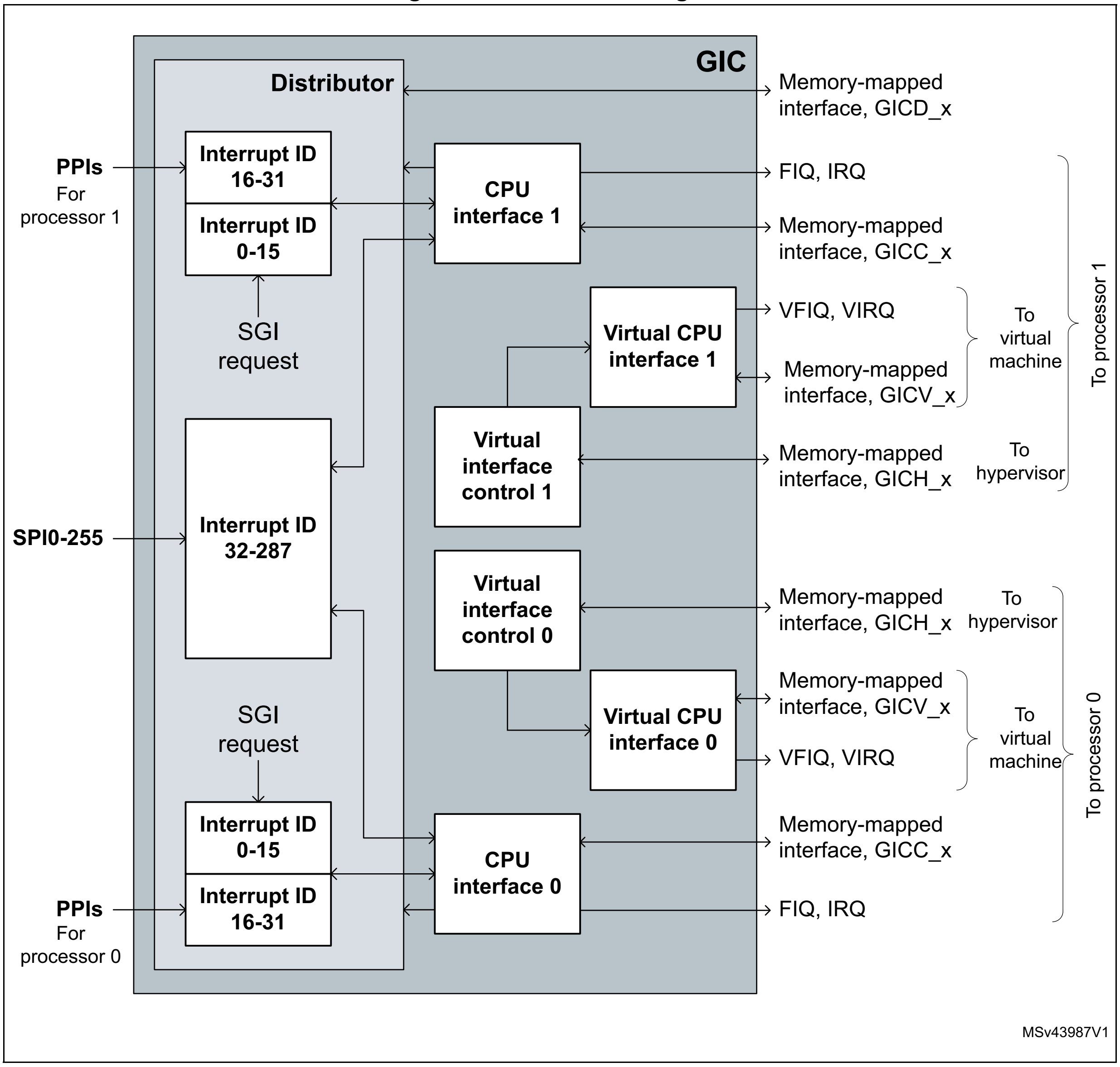
The GIC registers are memory-mapped, and the base address is visible into the configuration base address (CBAR) for each processor in the cluster.

Memory regions used for these registers must be marked as device or strongly-ordered in the translation tables.

Memory regions marked as normal memory cannot access any of the GIC registers, instead access caches or external memory as required.

Figure 136 shows the GIC block diagram.

Figure 136. GIC block diagram



22.4 Interrupt sources

All interrupt sources are identified by a unique ID.

The Cortex-A7 MPCore processor has the following interrupt sources:

22.4.1 Software generated interrupts (SGI)

SGIs are generated by writing to the software generated interrupt register (GICD_SGIR). A maximum of 16 SGIs, ID0-ID15, can be generated for each processor interface. An SGI has edge-triggered properties. The software triggering of the interrupt is equivalent to the edge transition of the interrupt signal on a peripheral input.

Arm recommends to use SGI ID0-ID7 for non-secure interrupts and SGI ID8-ID15 for secure interrupts.

22.4.2 Private peripheral interrupts (PPI)

A PPI is an interrupt generated by a peripheral that is specific to a single processor. There are seven PPIs for each CPU interface:

Legacy nFIQ signal (PPI0) - NOT USED

When the GIC interrupt bypass is in effect, such as after reset, the external nFIQ signal bypasses the interrupt distributor logic and directly drives the interrupt request to the corresponding processor.

When a processor uses the GIC rather than the external nFIQ signal, by enabling its own CPU interface, the nFIQ signal is treated like other interrupt lines and uses ID28. The interrupt is active-low level-sensitive.

Secure physical timer event (PPI1)

This is the event generated from the secure physical timer and uses ID29. The interrupt is level-sensitive.

Non-secure physical timer event (PPI2)

This is the event generated from the non-secure physical timer and uses ID30. The interrupt is level-sensitive.

Legacy nIRQ signal (PPI3) - NOT USED

When the GIC interrupt bypass is in effect, such as after reset, the external nIRQ signal bypasses the interrupt distributor logic and directly drives the interrupt request to the corresponding processor.

When a processor uses the GIC rather than the external nIRQ signal, by enabling its own CPU interface, the nIRQ signal is treated like other interrupt lines and uses ID31. The interrupt is active-low level-sensitive.

Virtual timer event (PPI4)

This is the event generated from the virtual timer and uses ID27. The interrupt is level-sensitive.

Hypervisor timer event (PPI5)

This is the event generated from the physical timer in hypervisor mode and uses ID26. The interrupt is level-sensitive.

Virtual maintenance interrupt (PPI6)

The virtualization extensions support in the Arm Generic Interrupt Controller Architecture Specification, permits for a maintenance interrupt to be generated under several conditions. The virtual maintenance interrupt uses ID25. The interrupt is level-sensitive.

22.4.3 Shared peripheral interrupts (SPI)

SPIs are triggered by events generated on associated interrupt input lines. The GIC supports up to 256 SPIs corresponding to the external IRQS signal.

SPIs start at ID32. The SPIs can be configured to be edge-triggered or active-high level-sensitive.

22.4.4 Interrupt priority formats

The Cortex-A7 MPCore processor implements a 5-bit version of the interrupt priority field for 32 interrupt priority levels.

22.5 GIC distributor (GICD)

The GIC distributor (GICD) centralizes all interrupt sources, determines the priority of each interrupt, and, for each CPU interface, forwards the interrupt with the highest priority to the interface for priority masking and preemption handling.

The GICD provides a programming interface for:

- globally enabling the forwarding of interrupts to the CPU interfaces
- enabling or disabling each interrupt
- setting the priority level of each interrupt
- setting the target processor list of each interrupt
- setting each peripheral interrupt to be level-sensitive or edge-triggered
- setting each interrupt as either group 0 or group 1
- sending an SGI to one or more target processors
- visibility of the state of each interrupt
- a mechanism for software to set or clear the pending state of a peripheral interrupt.

The GICD_IPRIORITYRx, GICD_ITARGETSRx, GICD_CPENDSGIRx, and GICD_SPENDSGIRx registers are byte-accessible and halfword-accessible.

All other registers are word-accessible.

22.5.1 GICD control register (GICD_CTLR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLEGRP1	ENABLEGRP0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **ENABLEGRP1:** enable group 1 interrupts
Global enable for forwarding pending group 1 interrupts from the GICD to the CPU interfaces

Bit 0 **ENABLEGRP0**: enable group 0 interrupts
Global enable for forwarding pending group 0 interrupts from the GICD to the CPU interfaces

22.5.2 GICD control non-secure access register (GICD_CTLRNS)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
Global enable for forwarding pending group 1 interrupts from the GICD to the CPU interfaces

22.5.3 GICD interrupt controller type register (GICD_TYPER)

Address offset: 0x004

Reset value: 0x0000 FC28

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LSPI[4:0]					SECURITYEXTN	Res.	Res.	CPUNUMBER[2:0]			ITLINESNUMBER[4:0]				
r	r	r	r	r	r			r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **LSPI[4:0]**: lockable shared peripheral interrupt
Returns the number of LSPIs that the interrupt controller contains (0b1111 = 31 LSPIs, these are the interrupts of IDs 32-62).

Bit 10 **SECURITYEXTN**: security extension
Indicates whether the GIC implements the security extensions. This bit always returns a value of 1, indicating that the security extensions are implemented.

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:5 **CPUNUMBER[2:0]**: number of processors interfaces
Indicates the number of implemented processors interfaces in the GIC (0b001 = 2 processors)

Bits 4:0 **ITLINESNUMBER[4:0]**: number of interrupt lines
Indicates the number of interrupts that the GIC supports (0b01000 = Up to 288 interrupts, 256 external interrupts)

22.5.4 GICD implementer identification register (GICD_IIDR)

Address offset: 0x008

Reset value: 0x0100 143B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCTID[7:0]								Res.	Res.	Res.	Res.	VARIANT[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISION[3:0]				IMPLEMENTER[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PRODUCTID[7:0]**: product ID of the GIC

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **VARIANT[3:0]**: Indicates the major revision number of the GIC

Bits 15:12 **REVISION[3:0]**: Indicates the minor revision number of the GIC

Bits 11:0 **IMPLEMENTER[11:0]**: GIC implementer (0x43B Arm implementation)

22.5.5 GICD interrupt group register x (GICD_IGROUPRx)

Address offset: 0x080 + 0x4 * x, (x = 0 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IGROUPRx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IGROUPRx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IGROUPRx[31:0]**: group of interrupts

22.5.6 GICD interrupt set-enable register (GICD_ISENABLER0)

Address offset: 0x100

Reset value: 0b0000 000x xxxx xxxx 1111 1111 1111 1111

For interrupts ID = 0 to ID = 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISENABLER0[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISENABLER0[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISENABLER0[31:0]**: interrupt set-enable 0

22.5.7 GICD interrupt set-enable register x (GICD_ISENABLERx)

Address offset: 0x104 + 0x4 * (x-1), (x = 1 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISENABLERx[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISENABLERx[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISENABLERx[31:0]:** interrupt set-enable x

22.5.8 GICD interrupt clear-enable register (GICD_ICENABLER0)

Address offset: 0x180

Reset value: 0b0000 000x xxxx xxxx 1111 1111 1111 1111

For interrupts ID = 0 to ID = 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICENABLER0[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICENABLER0[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICENABLER0[31:0]**: interrupt clear-enable 0

22.5.9 GICD interrupt clear-enable register x (GICD_ICENABLERx)

Address offset: 0x184 + 0x4 * (x-1), (x = 1 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICENABLERx[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICENABLERx[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICENABLERx[31:0]**: interrupt clear-enable x

22.5.10 GICD interrupt set-pending register x (GICD_ISPENDRx)

Address offset: 0x200 + 0x4 * x, (x = 0 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISPENDRx[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISPENDRx[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISPENDRx[31:0]**: interrupt set-pending x

22.5.11 GICD interrupt clear-pending register x (GICD_ICPENDRx)

Address offset: 0x280 + 0x4 * x, (x = 0 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICPENDRx[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICPENDRx[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICPENDRx[31:0]**: interrupt clear-pending x

22.5.12 GICD interrupt set-active register x (GICD_ISACTIVERx)

Address offset: 0x300 + 0x4 * x, (x = 0 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISACTIVERx[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISACTIVERx[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **ISACTIVERx[31:0]**: interrupt set-active x

22.5.13 GICD interrupt clear-active register x (GICD_ICACTIVERx)

Address offset: 0x380 + 0x4 * x, (x = 0 to 8)

Reset value: 0x0000 0000

For interrupts ID = x * 32 to ID = x * 32 + 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICACTIVERx[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICACTIVERx[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **ICACTIVERx[31:0]**: interrupt clear-active x

22.5.14 GICD interrupt priority register x (GICD_IPRIORITYRx)

Address offset: 0x400 + 0x4 * x, (x = 0 to 71)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIORITY3[4:0]					Res.	Res.	Res.	PRIORITY2[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIORITY1[4:0]					Res.	Res.	Res.	PRIORITY0[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw			

Bits 31:27 **PRIORITY3[4:0]:** priority for interrupt ID = x * 4 + 3

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:19 **PRIORITY2[4:0]:** priority for interrupt ID = x * 4 + 2

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:11 **PRIORITY1[4:0]:** priority for interrupt ID = x * 4 + 1

Bits 10:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY0[4:0]:** priority for interrupt ID = x * 4

Bits 2:0 Reserved, must be kept at reset value.

22.5.15 GICD interrupt processor target register x (GICD_ITARGETSRx)

Address offset: 0x800 + 0x4 * x, (x = 0 to 7)

Reset value: 0xXXXX XXXX

For existing SGIs and PPIs, read of CPU targets field returns the number of the processor performing the read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS2[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS0[1:0]	
						r	r							r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **CPU_TARGETS3[1:0]**: CPU(s) target for interrupt ID = x * 4 + 3

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **CPU_TARGETS2[1:0]**: CPU(s) target for interrupt ID = x * 4 + 2

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CPU_TARGETS1[1:0]**: CPU(s) target for interrupt ID = x * 4 + 1

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **CPU_TARGETS0[1:0]**: CPU(s) target for interrupt ID = x * 4

22.5.16 GICD interrupt processor target register x (GICD_ITARGETSRx)

Address offset: 0x820 + 0x4 * (x-8), (x = 8 to 71)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS2[1:0]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS0[1:0]	
						rw	rw							rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **CPU_TARGETS3[1:0]:** CPU(s) target for interrupt ID = x * 4 + 3

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **CPU_TARGETS2[1:0]:** CPU(s) target for interrupt ID = x * 4 + 2

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CPU_TARGETS1[1:0]**: CPU(s) target for interrupt ID = x * 4 + 1

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **CPU_TARGETS0[1:0]**: CPU(s) target for interrupt ID = x * 4

22.5.17 GICD interrupt configuration register (GICD_ICFGR0)

Address offset: 0xC00

Reset value: 0xAAAA AAAA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **INT_CONFIG15[1:0]**: interrupt config for interrupt ID = 15

Bits 29:28 **INT_CONFIG14[1:0]**: interrupt config for interrupt ID = 14

Bits 27:26 **INT_CONFIG13[1:0]**: interrupt config for interrupt ID = 13

Bits 25:24 **INT_CONFIG12[1:0]**: interrupt config for interrupt ID = 12

Bits 23:22 **INT_CONFIG11[1:0]**: interrupt config for interrupt ID = 11

Bits 21:20 **INT_CONFIG10[1:0]**: interrupt config for interrupt ID = 10

Bits 19:18 **INT_CONFIG9[1:0]**: interrupt config for interrupt ID = 9

Bits 17:16 **INT_CONFIG8[1:0]**: interrupt config for interrupt ID = 8

Bits 15:14 **INT_CONFIG7[1:0]**: interrupt config for interrupt ID = 7

Bits 13:12 **INT_CONFIG6[1:0]**: interrupt config for interrupt ID = 6

Bits 11:10 **INT_CONFIG5[1:0]**: interrupt config for interrupt ID = 5

Bits 9:8 **INT_CONFIG4[1:0]**: interrupt config for interrupt ID = 4

Bits 7:6 **INT_CONFIG3[1:0]**: interrupt config for interrupt ID = 3

Bits 5:4 **INT_CONFIG2[1:0]**: interrupt config for interrupt ID = 2

Bits 3:2 **INT_CONFIG1[1:0]**: interrupt config for interrupt ID = 1

Bits 1:0 **INT_CONFIG0[1:0]**: interrupt config for interrupt ID = 0

22.5.18 GICD interrupt configuration register (GICD_ICFGR1)

Address offset: 0xC04

Reset value: 0b0101 0101 0101 01xx xxxx xxxx xxxx xxxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **INT_CONFIG15[1:0]**: interrupt config for interrupt ID = 31

Bits 29:28 **INT_CONFIG14[1:0]**: interrupt config for interrupt ID = 30

Bits 27:26 **INT_CONFIG13[1:0]**: interrupt config for interrupt ID = 29

Bits 25:24 **INT_CONFIG12[1:0]**: interrupt config for interrupt ID = 28

Bits 23:22 **INT_CONFIG11[1:0]**: interrupt config for interrupt ID = 27

Bits 21:20 **INT_CONFIG10[1:0]**: interrupt config for interrupt ID = 26

Bits 19:18 **INT_CONFIG9[1:0]**: interrupt config for interrupt ID = 25

Bits 17:16 **INT_CONFIG8[1:0]**: interrupt config for interrupt ID = 24

Bits 15:14 **INT_CONFIG7[1:0]**: interrupt config for interrupt ID = 23

Bits 13:12 **INT_CONFIG6[1:0]**: interrupt config for interrupt ID = 22

Bits 11:10 **INT_CONFIG5[1:0]**: interrupt config for interrupt ID = 21

Bits 9:8 **INT_CONFIG4[1:0]**: interrupt config for interrupt ID = 20

Bits 7:6 **INT_CONFIG3[1:0]**: interrupt config for interrupt ID = 19

Bits 5:4 **INT_CONFIG2[1:0]**: interrupt config for interrupt ID = 18

Bits 3:2 **INT_CONFIG1[1:0]**: interrupt config for interrupt ID = 17

Bits 1:0 **INT_CONFIG0[1:0]**: interrupt config for interrupt ID = 16

22.5.19 GICD interrupt configuration register x (GICD_ICFGRx)

Address offset: 0xC08 + 0x4 * (x-2), (x = 2 to 17)

Reset value: 0x5555 5555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **INT_CONFIG15[1:0]**: interrupt config for interrupt ID = x * 16 + 15

Bits 29:28 **INT_CONFIG14[1:0]**: interrupt config for interrupt ID = x * 16 + 14

Bits 27:26 **INT_CONFIG13[1:0]**: interrupt config for interrupt ID = x * 16 + 13

Bits 25:24 **INT_CONFIG12[1:0]**: interrupt config for interrupt ID = x * 16 + 12

Bits 23:22 **INT_CONFIG11[1:0]**: interrupt config for interrupt ID = x * 16 + 11

Bits 21:20 **INT_CONFIG10[1:0]**: interrupt config for interrupt ID = x * 16 + 10

Bits 19:18 **INT_CONFIG9[1:0]**: interrupt config for interrupt ID = x * 16 + 9

Bits 17:16 **INT_CONFIG8[1:0]**: interrupt config for interrupt ID = x * 16 + 8

Bits 15:14 **INT_CONFIG7[1:0]**: interrupt config for interrupt ID = x * 16 + 7

Bits 13:12 **INT_CONFIG6[1:0]**: interrupt config for interrupt ID = x * 16 + 6

Bits 11:10 **INT_CONFIG5[1:0]**: interrupt config for interrupt ID = x * 16 + 5

Bits 9:8 **INT_CONFIG4[1:0]**: interrupt config for interrupt ID = x * 16 + 4

Bits 7:6 **INT_CONFIG3[1:0]**: interrupt config for interrupt ID = x * 16 + 3

Bits 5:4 **INT_CONFIG2[1:0]**: interrupt config for interrupt ID = x * 16 + 2

Bits 3:2 **INT_CONFIG1[1:0]**: interrupt config for interrupt ID = x * 16 + 1

Bits 1:0 **INT_CONFIG0[1:0]**: interrupt config for interrupt ID = x * 16

22.5.20 GICD private peripheral interrupt status register (GICD_PPISR)

Address offset: 0xD00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PPI3	PPI2	PPI1	PPI0	PPI4	PPI5	PPI6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r									

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PPI3**: nIRQ (not used)

Bit 14 **PPI2**: secure physical timer event

Bit 13 **PPI1**: secure physical timer event

Bit 12 **PPI0**: nFIQ (not used)

Bit 11 **PPI4**: virtual timer event

Bit 10 **PPI5**: hypervisor timer event

Bit 9 **PPI6**: virtual maintenance interrupt

Bits 8:0 Reserved, must be kept at reset value.

22.5.21 GICD shared peripheral interrupt register x (GICD_SPISRx)

Address offset: 0xD04 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

For interrupts ID = SPI number + 32, from SPI [x * 32 + 31] to SPI [x * 32]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPISRx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPISRx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SPISRx[31:0]**: shared peripheral interrupt

22.5.22 GICD software generated interrupt register (GICD_SGIR)

Address offset: 0xF00

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TARGETLISTFILTER[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPUTARGETLIST[1:0]	
						w	w							w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NSATT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SGIINTID[3:0]			
w												w	w	w	w

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **TARGETLISTFILTER[1:0]**: target list filter

Determines how the distributor must process the requested SGI

0: Forward the interrupt to the CPU interfaces specified in the CPUTARGETLIST field

1: Forward the interrupt to all CPU interfaces except the processor which requested the interrupt

2: Forward the interrupt only to the CPU interface of the processor which requested the interrupt

3: Reserved, must be kept at reset value.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **CPUTARGETLIST[1:0]**: CPU target list

When TARGETLISTFILTER = 0b00, defines the CPU interfaces to which the GICD must forward the interrupt. Each bit of CPUTARGETLIST[1:0] refers to the corresponding CPU interface. If this field is 0 when TARGETLISTFILTER is 0b00, the GICD does not forward the interrupt to any CPU interface.

Bit 15 **NSATT**: non-secure attribute

It specifies the required security value of the SGI:

0 - Forward the SGI specified in the SGIINTID field to a specified CPU interface only if the SGI is configured as group 0 on that interface.

1 - Forward the SGI specified in the SGIINTID field to a specified CPU interfaces only if the SGI is configured as group 1 on that interface.

This field is writable only by a secure access. Any non-secure write to the GICD_SGIR generates an SGI only if the specified SGI is programmed as group 1.

Bits 14:4 Reserved, must be kept at reset value.

Bits 3:0 **SGIINTID[3:0]**: SGI interrupt ID

The value of this field is the interrupt ID of the SGI to forward to the specified CPU interfaces, in the range 0-15 (for example: a value of 0b0011 specifies interrupt ID 3).

22.5.23 GICD SGI clear-pending register x (GICD_CPENDSGIRx)

Address offset: 0xF10 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

For SGI x * 4 to SGI x * 4 + 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING2[1:0]	
						rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING0[1:0]	
						rw	rw							rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **SGI_CLEAR_PENDING3[1:0]**: clear-pending state for SGI [x * 4 + 3]
Writing a 1 clears the pending state of the SGI [x * 4 + 3] for the corresponding source processor, and no longer targets the processor performing the write.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4 + 3] is pending, from the corresponding source processor, on the reading processor.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **SGI_CLEAR_PENDING2[1:0]**: clear-pending state for SGI [x * 4 + 2]
Writing a 1 clears the pending state of the SGI [x * 4 + 2] for the corresponding source processor, and no longer targets the processor performing the write.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4 + 2] is pending, from the corresponding source processor, on the reading processor.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **SGI_CLEAR_PENDING1[1:0]**:clear-pending state for SGI [x * 4 + 1]
Writing a 1 clears the pending state of the SGI [x * 4 + 1] for the corresponding source processor, and no longer targets the processor performing the write.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4 + 1] is pending, from the corresponding source processor, on the reading processor.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **SGI_CLEAR_PENDING0[1:0]**: clear-pending state for SGI [x * 4]
Writing a 1 clears the pending state of the SGI [x * 4] for the corresponding source processor, and no longer targets the processor performing the write.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4] is pending, from the corresponding source processor, on the reading processor.

22.5.24 GICD SGI set-pending register x (GICD_SPENDSGIRx)

Address offset: 0xF20 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

For SGI x * 4 to SGI x * 4 + 3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING2[1:0]	
						rw	rw							rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING0[1:0]	
						rw	rw							rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **SGI_SET_PENDING3[1:0]**: set-pending state for SGI [x * 4 + 3]
Writing a 1 sets the pending state of the SGI [x * 4 + 3] for the corresponding source processor.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4 + 3] is pending, from the corresponding source processor, on the reading processor.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **SGI_SET_PENDING2[1:0]**: set-pending state for SGI [x * 4 + 2]
Writing a 1 sets the pending state of the SGI [x * 4 + 2] for the corresponding source processor.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4 + 2] is pending, from the corresponding source processor, on the reading processor.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **SGI_SET_PENDING1[1:0]**: set-pending state for SGI [x * 4 + 1]
Writing a 1 sets the pending state of the SGI [x * 4 + 1] for the corresponding source processor.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4 + 1] is pending, from the corresponding source processor, on the reading processor.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **SGI_SET_PENDING0[1:0]**: set-pending state for SGI [x * 4]
Writing a 1 sets the pending state of the SGI [x * 4] for the corresponding source processor.
Writing a 0 has no effect.
Reading a bit identifies whether the SGI [x * 4] is pending, from the corresponding source processor, on the reading processor.

22.5.25 GICD peripheral ID4 register (GICD_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR4[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR4[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR4[31:0]**: peripheral ID4

22.5.26 GICD peripheral ID5 to ID7 register x (GICD_PIDRx)

Address offset: 0xFD4 + 0x4* (x-5), (x = 5 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDRx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDRx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDRx[31:0]**: peripheral ID5 to ID7

22.5.27 GICD peripheral ID0 register (GICD_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR0[31:0]**: peripheral ID0

22.5.28 GICD peripheral ID1 register (GICD_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR1[31:0]**: peripheral ID1

22.5.29 GICD peripheral ID2 register (GICD_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR2[31:0]**: peripheral ID2

22.5.30 GICD peripheral ID3 register (GICD_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR3[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR3[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR3[31:0]**: peripheral ID3

22.5.31 GICD component ID0 register (GICD_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR0[31:0]**: component ID0

22.5.32 GICD component ID1 register (GICD_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIDR1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR1[31:0]**: component ID1

22.5.33 GICD component ID2 register (GICD_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CIDR2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR2[31:0]**: component ID2

22.5.34 GICD component ID3 register (GICD_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDR3[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CIDR3[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CIDR3[31:0]**: component ID3

22.5.35 GICD register map

Table 120. GICD register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICD_CTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLEGRP1	ENABLEGRP0
	Reset value																															0	0
0x000	GICD_CTLRNS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0x004	GICD_TYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSPI[4:0]					SECURITYEXTN	Res.	Res.	CPUNUMBER[2:0]			ITLINESNUMBER[4:0]				
	Reset value																	1	1	1	1	1	1			0	0	1	0	1	0	0	0

Table 120. GICD register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	GICD_IIDR	PRODUCTID[7:0]								Res.	Res.	Res.	Res.	VARIANT[3:0]				REVISION[3:0]				IMPLEMENTER[11:0]											
	Reset value	0	0	0	0	0	0	0	1					0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
0x00C - 0x07C	Reserved	Reserved																															
0x080 to 0x0A0	GICD_IGROUPR0 to GICD_IGROUPR8	IGROUPRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 - 0x0FC	Reserved	Reserved																															
0x100	GICD_ISENABLER0	ISENABLER0[31:0]																															
	Reset value	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x104 to 0x120	GICD_ISENABLER1 to GICD_ISENABLER8	ISENABLERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124 - 0x17C	Reserved	Reserved																															
0x180	GICD_ICENABLER0	ICENABLER0[31:0]																															
	Reset value	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x184 to 0x1A0	GICD_ICENABLER1 to GICD_ICENABLER8	ICENABLERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A4 - 0x1FC	Reserved	Reserved																															
0x200 to 0x220	GICD_ISPENDR0 to GICD_ISPENDR8	ISPENDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x224 - 0x27C	Reserved	Reserved																															
0x280 to 0x2A0	GICD_ICPENDR0 to GICD_ICPENDR8	ICPENDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2A4 - 0x2FC	Reserved	Reserved																															
0x300 to 0x320	GICD_ISACTIVER0 to GICD_ISACTIVER8	ISACTIVERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x324 - 0x37C	Reserved	Reserved																															
0x380 to 0x3A0	GICD_ICACTIVER0 to GICD_ICACTIVER8	ICACTIVERx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3A4 - 0x3FC	Reserved	Reserved																															
0x400 to 0x51C	GICD_IPRIORITYR0 to GICD_IPRIORITYR71	PRIORITY3[4:0]					Res.	Res.	Res.	PRIORITY2[4:0]					Res.	Res.	Res.	PRIORITY1[4:0]					Res.	Res.	Res.	PRIORITY0[4:0]					Res.	Res.	Res.
	Reset value	0	0	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0	0			
0x520 - 0x7FC	Reserved	Reserved																															

Table 120. GICD register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x800 to 0x81C	GICD_ITARGETSR0 to GICD_ITARGETSR7	Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS2[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS0[1:0]	
	Reset value							x	x							x	x							x	x							x	x
0x820 to 0x91C	GICD_ITARGETSR8 to GICD_ITARGETSR71	Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS2[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPU_TARGETS0[1:0]	
	Reset value							0	0							0	0							0	0							0	0
0x920 - 0xBFC	Reserved	Reserved																															
0xC00	GICD_ICFGR0	INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]		INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
	Reset value	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0xC04	GICD_ICFGR1	INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]		INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
	Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xC08 to 0xC44	GICD_ICFGR2 to GICD_ICFGR17	INT_CONFIG15[1:0]		INT_CONFIG14[1:0]		INT_CONFIG13[1:0]		INT_CONFIG12[1:0]		INT_CONFIG11[1:0]		INT_CONFIG10[1:0]		INT_CONFIG9[1:0]		INT_CONFIG8[1:0]		INT_CONFIG7[1:0]		INT_CONFIG6[1:0]		INT_CONFIG5[1:0]		INT_CONFIG4[1:0]		INT_CONFIG3[1:0]		INT_CONFIG2[1:0]		INT_CONFIG1[1:0]		INT_CONFIG0[1:0]	
	Reset value	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0xC48 - 0xCFC	Reserved	Reserved																															
0xD00	GICD_PPISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPI3	PPI2	PPI1	PPI0	PPI4	PPI5	PPI6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0									
0xD04 to 0xD20	GICD_SPISR0 to GICD_SPISR7	SPISRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD24 - 0xEFC	Reserved	Reserved																															

Table 120. GICD register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF00	GICD_SGIR	Res.	Res.	Res.	Res.	Res.	Res.	TARGETLISTFILTER[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	CPUTARGETLIST[1:0]		NSATT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SGIINTID[3:0]			
	Reset value							x	x							x	x	x												x	x	x	x
0xF04 - 0xF0C	Reserved	Reserved																															
0xF10 to 0xF1C	GICD_CPENDSGIR0 to GICD_CPENDSGIR3	Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING2[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_CLEAR_PENDING0[1:0]	
	Reset value							0	0							0	0							0	0							0	0
0xF20 to 0xF2C	GICD_SPENDSGIR0 to GICD_SPENDSGIR3	Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING3[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING2[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING1[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	SGI_SET_PENDING0[1:0]	
	Reset value							0	0							0	0							0	0							0	0
0xF30 - 0xFCC	Reserved	Reserved																															
0xFD0	GICD_PIDR4	PIDR4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	GICD_PIDR5 to GICD_PIDR7	PIDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0	GICD_PIDR0	PIDR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
0xFE4	GICD_PIDR1	PIDR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0
0xFE8	GICD_PIDR2	PIDR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1
0xFEC	GICD_PIDR3	PIDR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFF0	GICD_CIDR0	CIDR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0xFF4	GICD_CIDR1	CIDR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0
0xFF8	GICD_CIDR2	CIDR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0xFFC	GICD_CIDR3	CIDR3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

22.6 GIC CPU Interface (GICC)

Each CPU interface provides a programming interface for:

- enabling the signaling of interrupt requests by the CPU interface
- acknowledging an interrupt
- indicating completion of the processing of an interrupt
- setting an interrupt priority mask for the processor
- defining the preemption policy for the processor
- determining the highest priority pending interrupt for the processor.

All the registers are word-accessible.

22.6.1 GICC control register (GICC_CTLR)

Address offset: 0x000

Reset value: 0bxxxx xxxx xxxx xxxx xxxx x000 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	EOIMODENS	EOIMODES	IRQBYPDISGRP1	FIQBYPDISGRP1	IRQBYPDISGRP0	FIQBYPDISGRP0	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **EOIMODENS**: Alias of EOIMODENS from the non-secure copy of this register.

Bit 9 **EOIMODES**:EOI mode for secure accesses

Controls the behavior of accesses to GICC_EOIR and GICC_DIR registers. This control applies only to secure accesses.

Bit 8 **IRQBYPDISGRP1**: Alias of IRQBYPDISGRP1 from the non-secure copy of this register.

Bit 7 **FIQBYPDISGRP1**: Alias of FIQBYPDISGRP1 from the non-secure copy of this register.

Bit 6 **IRQBYPDISGRP0**: IRQ bypass disable for group 0 interrupts

When the signaling of IRQs by the CPU interface is disabled, this bit partly controls whether the bypass IRQ signal is signaled to the processor.

Bit 5 **FIQBYPDISGRP0**: FIQ bypass disable for group 0 interrupts

When the signaling of FIQs by the CPU interface is disabled, this bit partly controls whether the bypass FIQ signal is signaled to the processor.

Bit 4 **CBPR**: BPR control

Controls whether the GICC_BPR provides common control to group 0 and group 1 interrupts.

Bit 3 **FIQEN**: FIQ enable for group 0 interrupts
Controls whether the CPU interface signals group 0 interrupts to a target processor using the FIQ or the IRQ signal.
The GIC always signals group 1 interrupts using the IRQ signal.

Bit 2 **ACKCTL**: Acknowledge control
When the highest priority pending interrupt is a group 1 interrupt, this bit determines both:
- whether a read of GICC_IAR acknowledges the interrupt, or returns a spurious interrupt ID
- whether a read of GICC_HPPIR returns the ID of the highest priority pending interrupt, or returns a spurious interrupt ID.
Arm deprecates use of this bit and strongly recommends using a software model where this bit is set to 0.

Bit 1 **ENABLEGRP1**: Enable group 1 interrupts
It enables for the signaling of group 1 interrupts by the CPU interface to the connected processor.

Bit 0 **ENABLEGRP0**: Enable group 0 interrupts
It enables for the signaling of group 0interrupts by the CPU interface to the connected processor.

22.6.2 GICC control non-secure access register (GICC_CTLRNS)

Address offset: 0x000

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xx0x x00x xxx0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EOIMODENS	Res.	Res.	IRQBYPDISGRP1	FIQBYPDISGRP1	Res.	Res.	Res.	Res.	ENABLEGRP1
						rw			rw	rw					rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **EOIMODENS**: EOI mode for non- secure accesses
It controls the behavior of accesses to GICC_EOIR and GICC_DIR registers.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **IRQBYPDISGRP1**: IRQ bypass for group 1 interrupts
When the signaling of IRQs by the CPU interface is disabled, this bit partly controls whether the bypass IRQ signal is signaled to the processor.

Bit 5 **FIQBYPDISGRP1**: FIQ bypass disable for group 1 interrupts
When the signaling of FIQs by the CPU interface is disabled, this bit partly controls whether the bypass FIQ signal is signaled to the processor.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **ENABLEGRP1**: Enable group1 interrupts
Enable for the signaling of group 1 interrupts by the CPU interface to the connected processor.

22.6.3 GICC input priority mask register (GICC_PMR)

Address offset: 0x004

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 0000 0xxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: priority mask level for the CPU interface
If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the processor.

Bits 2:0 Reserved, must be kept at reset value.

22.6.4 GICC binary point register (GICC_BPR)

Address offset: 0x008

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx x010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:
The value of this field controls how the 8-bit interrupt priority field is split into a group priority field. It is used to determine interrupt preemption and a sub-priority field. Minimum value is 2.

22.6.5 GICC binary point non-secure access register (GICC_BPRNS)

Address offset: 0x008

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx x011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:
The value of this field controls how the 8-bit interrupt priority field is split into a group priority field. It is used to determine interrupt preemption and a sub-priority field. Minimum value is 3.

22.6.6 GICC interrupt acknowledge register (GICC_IAR)

Address offset: 0x00C

Reset value: 0bxxxx xxxx xxxx xxxx xxxx x011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:
For SGIs in a multiprocessor implementation, this field identifies the processor that requested the interrupt. It returns the number of the CPU interface that made the request.
For example, a value of 1 means the request was generated by a write to the GICD_SGIR on CPU interface 1.

Bits 9:0 **INTERRUPT_ID[9:0]**: The interrupt ID

22.6.7 GICC end of interrupt register (GICC_EOIR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:
On a multiprocessor implementation, if the write refers to an SGI, this field contains the CPUID value from the corresponding GICC_IAR access.

Bits 9:0 **EOIINTID[9:0]**:
It contains the interrupt ID value from the corresponding GICC_IAR access.

22.6.8 GICC running priority register (GICC_RPR)

Address offset: 0x014

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								r	r	r	r	r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: current running priority on the CPU interface

Bits 2:0 Reserved, must be kept at reset value.

22.6.9 GICC highest priority pending interrupt register (GICC_HPPIR)

Address offset: 0x018

Reset value: 0bxxxx xxxx xxxx xxxx xxx0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the PENDINTID field returns the ID of an SGI, this field contains the CPUID value for that interrupt. This identifies the processor that generated the interrupt.

Bits 9:0 **PENDINTID[9:0]**: interrupt ID of the highest-priority pending interrupt

22.6.10 GICC aliased binary point register (GICC_ABPR)

Address offset: 0x01C

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx x011

GICC_ABPR is an alias of the non-secure GICC_BPR. When GICC_CTLR.CBPR is set to 0, a secure access to this register is equivalent to a non-secure access to GICC_BPR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field. It is used to determine interrupt preemption, and a subpriority field. Minimun value is 3.

22.6.11 GICC aliased interrupt acknowledge register (GICC_AIAR)

Address offset: 0x020

Reset value: 0bxxxx xxxx xxxx xxxx xxxx x011 1111 1111

GICC_AIAR is an alias of the non-secure view of GICC_IAR. A secure access to this register is identical to a non-secure access to GICC_IAR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

For SGIs in a multiprocessor implementation, this field identifies the processor that requested the interrupt. It returns the number of the CPU interface that made the request.

For example, a value of 1 means the request was generated by a write to the GICD_SGIR on CPU interface 1.

Bits 9:0 **INTERRUPT_ID[9:0]**: interrupt ID

22.6.12 GICC aliased end of interrupt register (GICC_AEOIR)

Address offset: 0x024

Reset value: 0xXXXX XXXX

GICC_AEOIR is an alias of the Non-secure GICC_EOIR. A secure access to this register is similar to a non-secure access to GICC_EOIR, except that the GICC_CTLR.EOImodeS bit is used.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, when processing an SGI, this field must contain the CPUID value from the corresponding GICC_AIAR, or non-secure GICC_IAR, access.

Bits 9:0 **EOIINTID[9:0]**:

It contains the interrupt ID value from the corresponding GICC_AIAR, or non-secure GICC_IAR, access.

22.6.13 GICC aliased highest priority pending interrupt register (GICC_AHPPIR)

Address offset: 0x028

Reset value: 0bxxxx xxxx xxxx xxxx xxx0 0011 1111 1111

ICC_AHPPIR is an alias of the non-secure GICC_HPPIR. A secure access to this register is equivalent to a non-secure access to GICC_HPPIR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the PENDINTID field returns the ID of an SGI, this field contains the CPUID value for that interrupt. This identifies the processor that generated the interrupt.

Bits 9:0 **PENDINTID[9:0]**: interrupt ID of the highest-priority pending interrupt

It contains the interrupt ID of the highest-priority pending interrupt If that interrupt is a group 1 interrupt. Otherwise, its value is the spurious interrupt ID, 1023.

22.6.14 GICC active priority register (GICC_APR0)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
APR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **APR0[31:0]:** active priority

22.6.15 GICC non-secure active priority register (GICC_NSAPR0)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSAPR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NSAPR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **NSAPR0[31:0]**: non-secure active priority

22.6.16 GICC interface identification register (GICC_IIDR)

Address offset: 0x0FC

Reset value: 0x0102 143B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCTID[11:0]												ARCH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REVISION[3:0]				IMPLEMENTER[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 **PRODUCTID[11:0]**: product ID

Bits 19:16 **ARCH[3:0]**: architecture version of the GIC

Bits 15:12 **REVISION[3:0]**: revision number for the CPU interface

Bits 11:0 **IMPLEMENTER[11:0]**: GIC implementer (0x43B Arm implementation)

22.6.17 GICC deactivate interrupt register (GICC_DIR)

Address offset: 0x1000

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:CPU ID
For an SGI in a multiprocessor implementation, this field identifies the processor that requested the interrupt

Bits 9:0 **INTERRUPT_ID[9:0]**: interrupt ID

22.6.18 GICC register map

Table 121. GICC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICC_CTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOIMODENS	EOIMODES	IRQBYPDISGRP1	FIQBYPDISGRP1	IRQBYPDISGRP0	FIQBYPDISGRP0	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x000	GICC_CTLRNS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOIMODENS	Res.	Res.	IRQBYPDISGRP1	FIQBYPDISGRP1	Res.	Res.	Res.	Res.	ENABLEGRP1
	Reset value																							0			0	0					0
0x004	GICC_PMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x008	GICC_BPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	0
0x008	GICC_BPRNS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	1
0x00C	GICC_IAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x010	GICC_EOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x

Table 121. GICC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	GICC_RPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									1	1	1	1	1			
0x018	GICC_HPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x01C	GICC_ABPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	1
0x020	GICC_AIAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x024	GICC_AEOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x028	GICC_AHPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x02C - 0x0CC	Reserved	Reserved																															
0x0D0	GICC_APR0	APR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4 - 0x0DC	Reserved	Reserved																															
0x0E0	GICC_NSAPR0	NSAPR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4 - 0x0F8	Reserved	Reserved																															
0x0FC	GICC_IIDR	PRODUCTID[11:0]												ARCH[3:0]				REVISION[3:0]				IMPLEMENTER[11:0]											
	Reset value	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
0x100 - 0xFFC	Reserved	Reserved																															
0x1000	GICC_DIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x

Refer to *Section 2.5 on page 153* for the register boundary addresses.

22.7 GIC virtual interface control, common (GICH)

The virtual interface control registers are management registers. Configuration software on the Cortex-A7 MPCore processor must ensure they are accessible only by a hypervisor or similar software.

All the registers are word-accessible.

22.7.1 GICH hypervisor control register (GICH_HCR)

Address offset: 0x000

Reset value: 0b0000 0xxx xxxx xxxx xxxx xxxx 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EOICOUNT[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1DIE	VGRP1EIE	VGRP0DIE	VGRP0EIE	NPIE	LRENPIE	UIE	EN
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **EOICOUNT[4:0]**: end-of-interrupt counter

Counts the number of EOIs received that do not have a corresponding entry in the list registers. The virtual CPU interface increments this field automatically when a matching EOI is received. When EOIs that do not clear a bit in the active priorities register, GICH_APR does not cause an increment.

Although not possible under correct operation, if an EOI occurs when the value of this field is 31, this field wraps to 0.

The maintenance interrupt is asserted whenever this field is non-zero and the LRENPIE bit is set to 1.

Bits 26:8 Reserved, must be kept at reset value.

Bit 7 **VGRP1DIE**: virtual machine disable group 1 interrupt enable

Enables the signaling of a maintenance interrupt while signaling of group 1 interrupts from the virtual CPU interface to the connected virtual machine, is disabled.

Bit 6 **VGRP1EIE**: virtual machine enable group 1 interrupt enable

Enables the signaling of a maintenance interrupt while signaling of group 1 interrupts from the virtual CPU interface to the connected virtual machine, is enabled.

Bit 5 **VGRP0DIE**: virtual machine disable group 0interrupt enable

Enables the signaling of a maintenance interrupt while signaling of group 0interrupts from the virtual CPU interface to the connected virtual machine, is disabled.

Bit 4 **VGRP0EIE**: virtual machine enable group 0interrupt enable

Enables the signaling of a maintenance interrupt while signaling of group 0interrupts from the virtual CPU interface to the connected virtual machine, is enabled.

Bit 3 **NPIE**: no pending interrupt enable

Enables the signaling of a maintenance interrupt while no pending interrupts are present in the list registers.

Bit 2 **LRENPIE**: list register entry not present interrupt enable
Enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid list register entry for an EOI request.

Bit 1 **UIE**: underflow interrupt enable.
Enables the signaling of a maintenance interrupt when the list registers are empty, or hold only one valid entry.

Bit 0 **EN**: global enable bit for the virtual CPU interface

22.7.2 GICH VGIC type register (GICH_VTR)

Address offset: 0x004

Reset value: 0x9000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIBITS[2:0]			PREBITS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r										
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LISTREGS[4:0]				
											r	r	r	r	r

Bits 31:29 **PRIBITS[2:0]**: priority bits
Indicates the number of priority bits implemented, minus one.
For example, 0x4 means 5 bits of priority and 32 priority levels.

Bits 28:26 **PREBITS[2:0]**: preemption bits
Indicates the number of preemption bits implemented, minus one.
For example, 0x4 means 5 bits of preemption and 32 preemption levels

Bits 25:5 Reserved, must be kept at reset value.

Bits 4:0 **LISTREGS[4:0]**: list registers
Indicates the number of implemented list registers, minus one.
For example, 0x3 means 4 list registers.

22.7.3 GICH virtual machine control register (GICH_VMCR)

Address offset: 0x008

Reset value: 0b0000 0xxx 0100 11x1 xxxx xx0x xxx0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VMPRIMASK[4:0]					Res.	Res.	Res.	VMBP[2:0]			VMABP[2:0]			Res.	Res.
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	VEM	Res.	Res.	Res.	Res.	VMCBPR	VMFIQEN	VMACKCTL	VMGRP1EN	VMGRP0EN
						rw					rw	rw	rw	rw	rw

Bits 31:27 **VMPRIMASK[4:0]**: alias of GICV_PMR.PRIORITY

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:21 **VMBP[2:0]**: alias of GICV_BPR.BINARY_POINT

Bits 20:18 **VMABP[2:0]**: alias of GICV_ABPR.BINARY_POINT.

Bits 17:10 Reserved, must be kept at reset value.

Bit 9 **VEM**: alias of GICV_CTLR.EOIMODE

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **VMCBPR**: alias of GICV_CTLR.CBPR

Bit 3 **VMFIQEN**: alias of GICV_CTLR.FIQEN

Bit 2 **VMACKCTL**: alias of GICV_CTLR.ACKCTL

Bit 1 **VMGRP1EN**: alias of GICV_CTLR.ENABLEGRP1

Bit 0 **VMGRP0EN**: alias of GICV_CTLR.ENABLEGRP0

22.7.4 GICH maintenance interrupt status register (GICH_MISR)

Address offset: 0x010

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1D	VGRP1E	VGRP0D	VGRP0E	NP	LRENP	U	EOI
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **VGRP1D**: disabled group 1 maintenance interrupt
Asserted whenever GICH_HCR.VGRP1DIE is set and GICH_VMCR.VMGRP1EN=0.

Bit 6 **VGRP1E**: enabled group 1 maintenance interrupt
Asserted whenever GICH_HCR.VGRP1EIE is set and GICH_VMCR.VMGRP1EN=1.

Bit 5 **VGRP0D**: disabled group 0 maintenance interrupt
Asserted whenever GICH_HCR.VGRP0DIE is set and GICH_VMCR.VMGRP0EN=0.

Bit 4 **VGRP0E**: enabled group 0 maintenance interrupt
Asserted whenever GICH_HCR.VGRP0EIE is set and GICH_VMCR.VMGRP0EN=1.

Bit 3 **NP**: no pending maintenance interrupt
Asserted whenever GICH_HCR.NPIE=1 and no list register is in pending state.

Bit 2 **LRENP**: list register entry not present maintenance interrupt

Asserted whenever GICH_HCR.LRENPIE=1 and GICH_HCR.EOICOUNT is non-zero.

Bit 1 **U**: underflow maintenance interrupt

Asserted whenever GICH_HCR.UIE is set and if none, or only one, of the list register entries are marked as a valid interrupt, that is, if the corresponding GICH_LRx.STATE bits do not equal 0x0.

Bit 0 **EOI**: End of interrupt maintenance interrupt

Asserted whenever at least one list register is asserting an EOI Interrupt., which means at least one bit in GICH_EISRx=1.

22.7.5 GICH end of interrupt status register (GICH_EISR0)

Address offset: 0x020

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EISR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EISR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EISR0[31:0]**: end of interrupt status

22.7.6 GICH empty list status register (GICH_ELSR0)

Address offset: 0x030

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ELSR0[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ELSR0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ELSR0[31:0]**: empty list status

22.7.7 GICH active priority register (GICH_APR0)

Address offset: 0x0F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
APR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **APR0[31:0]**: active priority

22.7.8 GICH list register x (GICH_LRx)

Address offset: 0x100 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HW	GRP1	STATE[1:0]		PRIORITY[4:0]					Res.	Res.	Res.	PHYSICALID[9:6]			
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PHYSICALID[5:0]						VIRTUALID[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **HW**: hardware interrupt

Indicates whether this virtual interrupt is a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the PhysicalID field indicates.

Bit 30 **GRP1**: Indicates whether this virtual interrupt is a group 1 virtual interrupt

Bits 29:28 **STATE[1:0]**: state of the interrupt

Bits 27:23 **PRIORITY[4:0]**: priority of the interrupt

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:10 **PHYSICALID[9:0]**: physical ID

The function of this bit depends on the value of the GICH_LR.HW bit, as follows.

1) When GICH_LR.HW is set to 0, bits [9:0] have the following meanings:

- [19] EOI when this interrupt triggers an EOI maintenance interrupt
- [12:10] CPUID If the interrupt has the VirtualID for an SGI, that is, 0-15. This field shows the requesting CPU ID. This appears in the relevant field of the virtual machine interrupt acknowledge register, GICV_IAR or GICV_AIAR. Otherwise, this field must be set to 0.

2) When GICH_LR.HW is set to 1, PHYSICALID[9:0] indicates the physical interrupt ID that the hypervisor forwards to the GICD.

If the value of PHYSICALID is 0-15, or 1020-1023, behavior is **unpredictable**.

If the value of PHYSICALID is 16-31, this field applies to the PPI associated with the same physical CPUID as the virtual CPU interface requesting the deactivation.

Bits 9:0 **VIRTUALID[9:0]**: virtual ID

This ID is returned to the guest OS when the interrupt is acknowledged through the VM interrupt acknowledge register, GICV_IAR. Each valid interrupt stored in the list registers must have a unique virtual ID for that virtual CPU interface.

If the value of VIRTUALID is 1020-1023, behavior is **unpredictable**.

22.7.9 GICH register map

Table 122. GICH register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICH_HCR	EOICOUNT[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1DIE	VGRP1EIE	VGRP0DIE	VGRP0EIE	NPIE	LRENPIE	UIE	EN
	Reset value	0	0	0	0	0																				0	0	0	0	0	0	0	0
0x004	GICH_VTR	PRIBITS[2:0]			PREBITS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LISTREGS[4:0]				
	Reset value	1	0	0	1	0	0																						0	0	0	1	1
0x008	GICH_VMCR	VMPRIMASK[4:0]					Res.	Res.	Res.	VMBP[2:0]			VMABP[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMCBPR	VMFIQEN	VMACKCTL	VMGRP1EN	VMGRP0EN
	Reset value	0	0	0	0	0				0	1	0	0	1	1														0	0	0	0	0
0x00C	Reserved	Reserved																															
0x010	GICH_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VGRP1D	VGRP1E	VGRP0D	VGRP0E	NP	LRENP	U	EOI
	Reset value																									0	0	0	0	0	0	0	0
0x014 - 0x01C	Reserved	Reserved																															
0x020	GICH_EISR0	EISR0[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0
0x024 - 0x02C	Reserved	Reserved																															

Table 122. GICH register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	GICH_ELSR0	ELSR0[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1	1
0x034 - 0x0EC	Reserved	Reserved																															
0x0F0	GICH_APR0	APR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F4 - 0x0FC	Reserved	Reserved																															
0x100 to 0x10C	GICH_LR0 to GICH_LR3	HW	GRP1	STATE[1:0]		PRIORITY[4:0]					Res.	Res.	Res.	PHYSICALID[9:0]										VIRTUALID[9:0]									
	Reset value	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

22.8 GIC virtual CPU interface (GICV)

The virtual CPU interface forwards virtual interrupts to a connected Cortex-A7 MPCore processor, subject to the normal GIC handling and prioritization rules. The virtual interface control registers control virtual CPU interface operation, and in particular, the virtual CPU interface uses the contents of the List registers to determine when to signal virtual interrupts.

When a processor accesses the virtual CPU interface, the list registers are updated.

All the registers are word-accessible.

22.8.1 GICV virtual machine control register (GICV_CTLR)

Address offset: 0x000

Reset value: 0bxxxx xxxx xxxx xxxx xxxx x000 0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	EOIMODE	Res.	Res.	Res.	Res.	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
						rw					rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **EOIMODE**: end of interrupt mode
Controls the behavior associated with the GICV_EOIR, GICV_AEOIR, and GICV_DIR registers.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **CBPR**: BPR control
Controls whether the GICV_BPR controls both group 0 and group 1 virtual interrupts.

Bit 3 **FIQEN**: FIQ enable for group 0 interrupts
Controls whether interrupts marked as group 0 are presented as virtual FIQs.

Bit 2 **ACKCTL**: acknowledge control
Controls whether a read of the GICV_IAR, when the highest priority pending interrupt is a group 1 interrupt, causes the CPU interface to acknowledge the interrupt.
Arm deprecates use of this bit. Arm strongly recommends that software is written to operate with this bit always set to 0.

Bit 1 **ENABLEGRP1**:
Enables the signaling of group 1 virtual interrupts by the virtual CPU interface to the virtual machine.

Bit 0 **ENABLEGRP0**:
Enables the signaling of group 0 virtual interrupts by the virtual CPU interface to the virtual machine.

22.8.2 GICV VM priority mask register (GICV_PMR)

Address offset: 0x004

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 0000 0xxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: priority mask level for the virtual CPU interface
Only virtual interrupts with higher priority than the value in this register can be signaled to the processor

Bits 2:0 Reserved, must be kept at reset value.

22.8.3 GICV VM binary point register (GICV_BPR)

Address offset: 0x008

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx x010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:
The value of this field controls how the 8-bit virtual interrupt priority field is split into a group priority field, used to determine virtual interrupt preemption, and a subpriority field. Minimun value is 2.

22.8.4 GICV VM interrupt acknowledge register (GICV_IAR)

Address offset: 0x00C

Reset value: 0bxxxx xxxx xxxx xxxx xxx0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:
If the GICH_LRx.HW bit is set to 0, indicating that the interrupt is triggered in software, then CPUID of the GICH_LRx, that indicate the CPU ID, are returned in the GICV_IAR.CPUID field. Otherwise GICV_IAR.CPUID field reads as zero.

Bits 9:0 **INTERRUPT_ID[9:0]**: The interrupt ID

22.8.5 GICV VM end of interrupt register (GICV_EOIR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the write refers to an SGI, this field contains the CPUID value from the corresponding GICV_IAR access.

Bits 9:0 **EOIINTID[9:0]**:

If the GICH_LRx.HW bit in the matching list register is set to 1, indicating a hardware interrupt, then a deactivate request is sent to the physical Distributor, identifying the Physical ID from the corresponding field in the list register.

22.8.6 GICV VM running priority register (GICV_RPR)

Address offset: 0x014

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
								r	r	r	r	r			

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:3 **PRIORITY[4:0]**: current running priority on the virtual CPU interface

Bits 2:0 Reserved, must be kept at reset value.

22.8.7 GICV VM highest priority pending interrupt register (GICV_HPPIR)

Address offset: 0x018

Reset value: 0bxxxx xxxx xxxx xxxx xxx0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**: On a multiprocessor implementation, if GICH_LRx.HW bit=0, this field contains the CPUID value for that virtual interrupt. This identifies the virtual processor that generated the virtual interrupt.

Bits 9:0 **PENDINTID[9:0]**: The virtual interrupt ID of the highest priority pending virtual interrupt

22.8.8 GICV VM aliased binary point register (GICV_ABPR)

Address offset: 0x01C

Reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx xxxx x011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **BINARY_POINT[2:0]**:

The value of this field controls how the 8-bit virtual interrupt priority field is split into a group priority field, used to determine virtual interrupt preemption, and a subpriority field. Minimun value is 3.

22.8.9 GICV VM aliased interrupt register (GICV_AIAR)

Address offset: 0x020

Reset value: 0bxxxx xxxx xxxx xxxx xxx0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**: If the GICH_LRx.HW bit is set to 0, indicating that the interrupt is triggered in software, then CPUID of the GICH_LRx, that indicate the CPU ID, are returned in the GICV_AIAR.CPUID field. Otherwise GICV_AIAR.CPUID field reads as zero..

Bits 9:0 **INTERRUPT_ID[9:0]**: The interrupt ID. If the GICH_LRx.Grp1 bit is 0, the interrupt is Group 0. The spurious interrupt ID 1023 is returned and the interrupt is not acknowledged.

22.8.10 GICV VM aliased end of interrupt register (GICV_AEOIR)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

On a multiprocessor implementation, if the write refers to an SGI, this field contains the CPUID value from the corresponding GICV_IAR access.

Bits 9:0 **EOIINTID[9:0]**:

If the GICH_LRx.HW bit in the matching list register is set to 1, indicating a hardware interrupt, then a deactivate request is sent to the physical Distributor, identifying the Physical ID from the corresponding field in the list register.

22.8.11 GICV VM aliased highest priority pending interrupt register (GICV_AHPPIR)

Address offset: 0x028

Reset value: 0bxxxx xxxx xxxx xxxx xxx0 0011 1111 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**:

If the GICH_LRx.HW bit in the matching List register is set to 1, indicating a hardware interrupt, then a deactivate request is sent to the physical Distributor, identifying the physical ID from the corresponding field in the List register.

Bits 9:0 **PENDINTID[9:0]**:

The virtual interrupt ID of the highest-priority pending virtual interrupt, if that virtual interrupt is a group 1 virtual interrupt. Otherwise, the spurious virtual interrupt ID, 1023.

22.8.12 GICV VM active priority register (GICV_APR0)

Address offset: 0x0D0

Reset value: 0x0000 0000

The GICV_APR0 is an alias of GICH_APR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APR0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APR0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **APR0[31:0]:** active priority

22.8.13 GICV VM CPU interface identification register (GICV_IIDR)

Address offset: 0x0FC

Reset value: 0x0102 143B

The GICV_IIDR is an alias of GICC_IIDR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IIDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IIDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IIDR[31:0]:** see description of GICC_IIDR register

22.8.14 GICV VM deactivate interrupt register (GICV_DIR)

Address offset: 0x1000

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
					w	w	w	w	w	w	w	w	w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CPUID**: This field identifies the processor that requested the interrupt.

Bits 9:0 **INTERRUPT_ID[9:0]**: The interrupt ID

22.8.15 GICV register map

Table 123. GICV register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GICV_CTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EOIMODE	Res.	Res.	Res.	Res.	CBPR	FIQEN	ACKCTL	ENABLEGRP1	ENABLEGRP0
	Reset value																							0					0	0	0	0	0
0x004	GICV_PMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x008	GICV_BPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	0
0x00C	GICV_IAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x010	GICV_EOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x014	GICV_RPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIORITY[4:0]					Res.	Res.	Res.
	Reset value																									1	1	1	1	1			
0x018	GICV_HPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x01C	GICV_ABPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BINARY_POINT[2:0]		
	Reset value																														0	1	1
0x020	GICV_AIAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x024	GICV_AEOIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	EOIINTID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x
0x028	GICV_AHPPIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	PENDINTID[9:0]									
	Reset value																						0	1	1	1	1	1	1	1	1	1	1
0x02C - 0x0CC	Reserved	Reserved																															

Table 123. GICV register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D0	GICV_APR0	APR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4 - 0x0F8	Reserved	Reserved																															
0x0FC	GICV_IIDR	IIDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1
0x100 - 0xFFC	Reserved	Reserved																															
0x1000	GICV_DIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUID	INTERRUPT_ID[9:0]									
	Reset value																						x	x	x	x	x	x	x	x	x	x	x

Refer to *Section 2.5 on page 153* for the register boundary addresses.

23 Extended interrupt and event controller (EXTI)

The Extended interrupt and event controller (EXTI) manages the individual CPU and system wakeup through configurable and direct event inputs. It provides wakeup requests to the power control, and generates an interrupt request to the CPUs interrupt controller and events to the CPUs event input. For each CPU an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wakeup requests allow the system to be woken up from STOP modes, and the CPU to be woken up from the CSTOP and CSTANDBY modes.

The interrupt request and event request generation can also be used in RUN modes.

The EXTI also includes the EXTI mux IOport selection.

CPU1 = MPU and CPU2 = MCU

23.1 EXTI main features

The EXTI main features are the following:

- 76 input events supported
- Two CPUs supported
- All event inputs allow to wake up the system
- Events which do not have an associated wakeup flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI
- Some events can be used to generate a CPU wakeup event

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse)
  – Configurable events have the following features:
    - Selectable active trigger edge
    - Interrupt pending status register bit independent for the rising and falling edge.
    - Individual interrupt and event generation mask, used for conditioning the CPU wakeup, interrupt and event generation
    - SW trigger possibility
- Direct events (interrupt and wakeup sources from peripherals having an associated flag which requiring to be cleared in the peripheral)
  – Direct events have the following features:
    - Fixed rising edge active trigger
    - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
    - Individual interrupt and event generation mask, used for conditioning the CPU wakeup and event generation
    - No SW trigger possibility
- TrustZone secure events
  – The access to control and configuration bits of secure input events can be made secure.
- EXTI IO port selection

23.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input Trigger block, the masking block, and EXTI mux as shown in *Figure 137*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and the masking of these.

The EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 137. EXTI block diagram

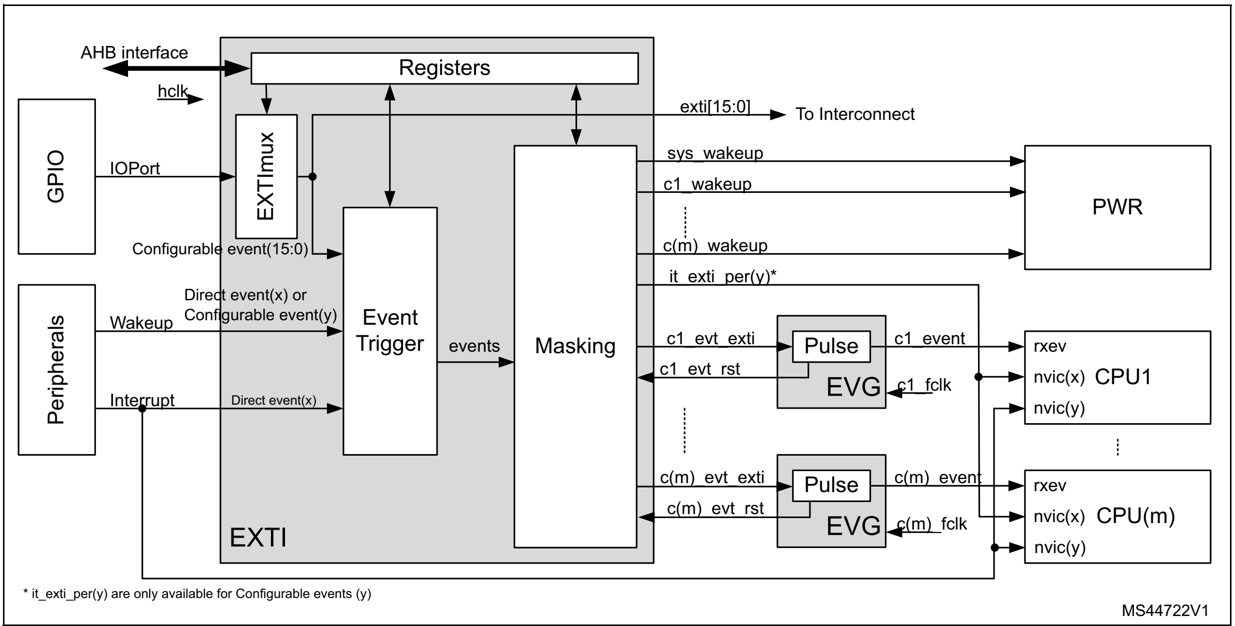


Table 124. EXTI pin overview

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to allow security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock.
Configurable event(y)	I	Asynchronous wakeup events from peripherals that do not have an associated interrupt and flag in the peripheral.
Direct event(x)	I	Synchronous and asynchronous wakeup events from peripherals having an associated interrupt and flag in the peripheral.
IOPort(n)	I	GPIOs block IO ports[15:0].
exti[15:0]	O	EXTI output port to trigger other IPs.
it_exti_per (y)	O	Interrupts to the CPU1 to CPU(m) associated with Configurable event (y).
c(m)_evt_exti	O	High level sensitive event output for CPU(m) synchronous to hclk. (m= 2)
c(m)_evt_rst	I	Asynchronous reset input to clear c(m)_evt_exti. (m= 2)

Table 124. EXTI pin overview (continued)

Pin name	I/O	Description
sys_wakeup	O	Asynchronous system wakeup request to PWR for ck_sys and hclk.
c(m)_wakeup	O	Wakeup request to PWR for CPU(m), synchronous to hclk. (m= 2)

Table 125. EVG pin overview

Pin name	I/O	Description
c_fclk	I	CPU free running clock.
c_evt_in	I	High level sensitive Events input from EXTI, asynchronous to CPU clock.
c_event	O	Event pulse, synchronous to CPU clock.
c_evt_rst	O	Event reset signal, synchronous to CPU clock.

23.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup or interrupt events when the system is in STOP mode or a CPU is in CSTOP mode are connected to the EXTI.

- Peripheral wakeup signals that generate a pulse or which do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events the EXTI provides a status pending bit that has to be cleared. It is the EXTI interrupt associated with the status bit that will interrupt the CPU.
- Peripheral interrupt and wakeup signals that have a status bit in the peripheral that has to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral. It is the peripheral interrupt that will interrupt the CPU directly.
- All GPIO ports input to the EXTI multiplexer, allowing to select a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the respective interrupt controller of each CPU(m).

The dedicated EXTI/EVG CPU(m) event is connected to the respective CPU(m) rxev input.

The EXTI CPU(m) wakeup signals are connected to the PWR block, and are used to wake up the system and CPU(m) sub-system bus clocks.

23.3 EXTI functional description

Depending on the EXTI event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by rising edge selection
  EXTI rising trigger selection register (EXTI_RTSR1),
  EXTI rising trigger selection register (EXTI_RTSR2),
  EXTI rising trigger selection register (EXTI_RTSR3),
  and falling edge selection
  EXTI falling trigger selection register (EXTI_FTSR1),
  EXTI falling trigger selection register (EXTI_FTSR2),
  EXTI falling trigger selection register (EXTI_FTSR3).
- Software trigger, by
  EXTI software interrupt event register (EXTI_SWIER1),
  EXTI software interrupt event register (EXTI_SWIER2),
  EXTI software interrupt event register (EXTI_SWIER3).
- Interrupt pending flag, by
  EXTI rising edge pending register (EXTI_RPR1),
  EXTI rising edge pending register (EXTI_RPR2)
  EXTI rising edge pending register (EXTI_RPR3)
  EXTI falling edge pending register (EXTI_FPR2)
  EXTI falling edge pending register (EXTI_FPR3).
- CPU wakeup and interrupt enable, by
  EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR1),
  EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1),
  EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR2),
  EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2),
  EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR3)
  EXTI CPUm wakeup with interrupt mask register (EXTI_C2IMR3).
- CPU wakeup and event enable, by
  EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1)
  EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2),
  EXTI CPU2 wakeup with event mask register (EXTI_C2EMR3).

Table 126. EXTI event input configurations and register control

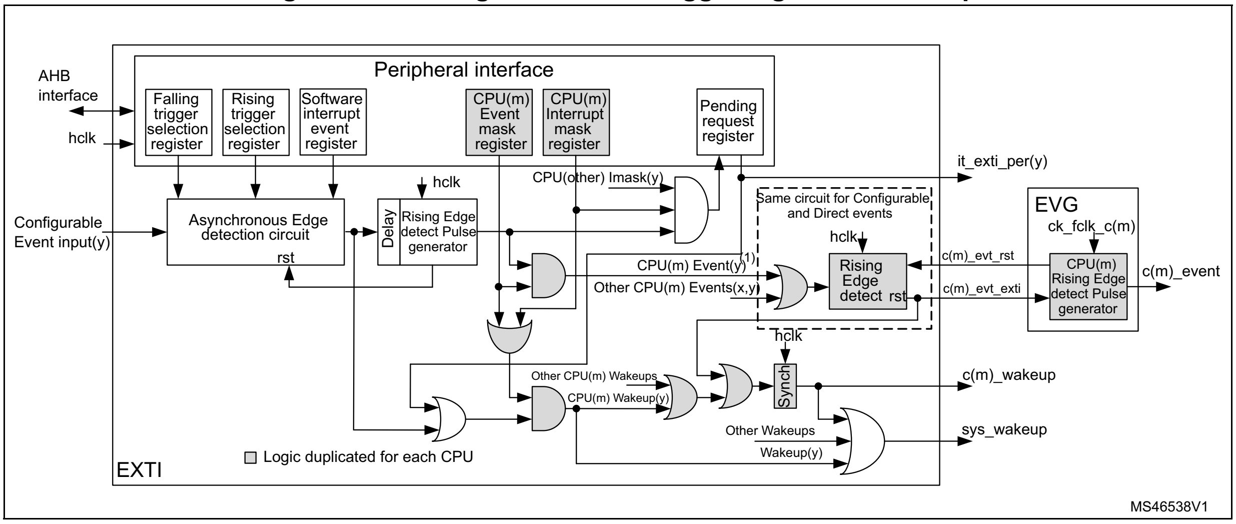
Event input type	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_R/FPR	EXTI_CmIMR	EXTI_CmEMR(1)
Configurable	Configurable event input wakeup logic	x	x	x	x	x	x
Direct	Direct event input wakeup logic	-	-	-	-	x	x

1. Only for input events with configuration “rxev generation” enabled.

23.3.1 EXTI configurable event input wakeup

Figure 138 is a detailed representation of the logic associated with configurable event inputs which will wake up the CPU(m) sub-system bus clocks and generated an EXTI pending flag and interrupt to the CPU(m) and or a CPU(m) wakeup event.

Figure 138. Configurable event trigger logic CPU wakeup



1. Only for the input events that support CPU rxev generation c(n)_event.

The software interrupt event register allows to trigger configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

The rising edge and falling edge selection registers allow to enable and select the configurable event active trigger edge or both edges.

Each CPU has its dedicated interrupt mask register and a dedicated event mask registers. The enabled event allows to generate an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) is not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers, shared by the CPUs. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to all CPUs. The configurable event interrupts need to be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

When a CPU(m) interrupt or CPU(m) event is enabled the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event interrupt pending request, may be cleared by any CPU. The system is unable to enter into Low-power modes as long as an interrupt pending request is active.

23.3.2 EXTI direct event input wakeup

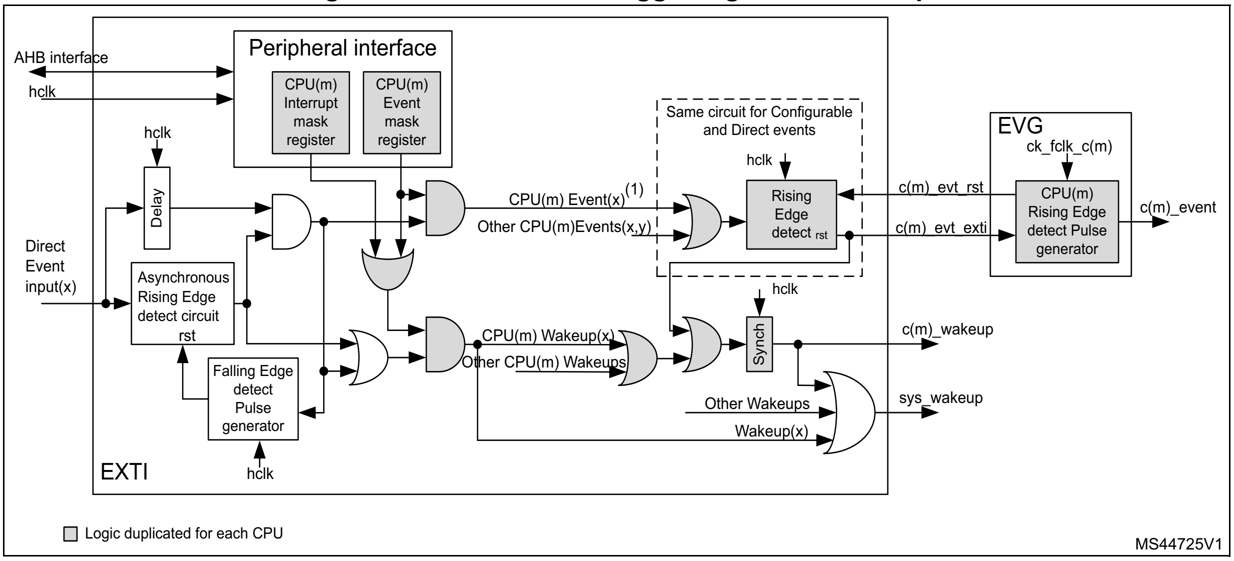
Figure 139 is a detailed representation of the logic associated with direct event inputs waking up the system.

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU sub-system clocks and may generate a CPU wakeup event. The peripheral synchronous interrupt, associated with the direct wakeup event wakes up the CPU.

The EXTI direct event is able to generate a CPU event. This CPU event wakes up the CPU. The CPU event may occur before the associated Peripheral interrupt flag is set.

Note: The direct events are cleared in the peripheral generating the event. When a direct event input enabled by CPU(m) is cleared by another CPU before the CPU(m) clock is running, the CPU(m) no longer receives a CPU(m) interrupt nor a CPU(m) event and does not wake up. However the system stays in RUN mode, generating the CPU(m) clock. For this reason CPU(m) direct events must NOT be cleared by the other CPU.

Figure 139. Direct event trigger logic CPU wakeup

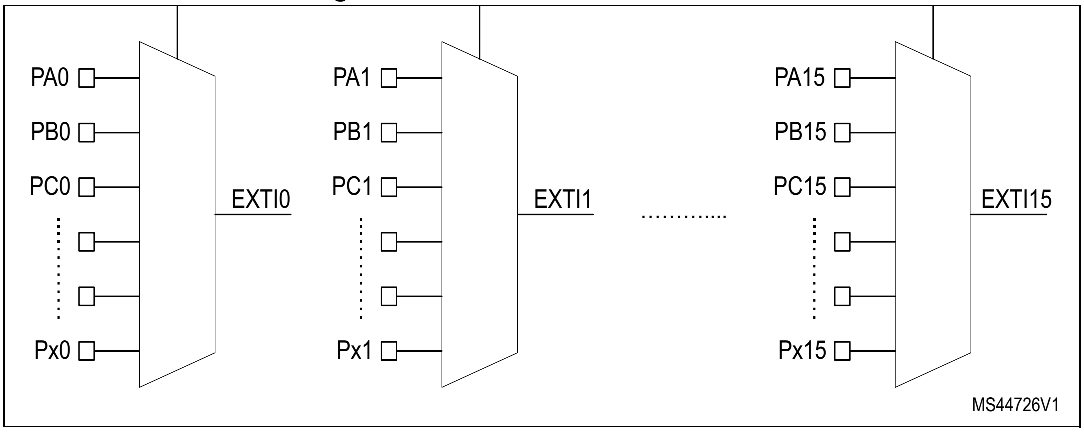


1. Only for the input events that support CPU rxev generation c(n)_event.

23.3.3 EXTI mux selection

The EXTI mux allows to select GPIOs as interrupts and wakeup. The GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output, is controlled by registers: *EXTI external interrupt selection register 1 (EXTI_EXTICR1)* to *EXTI external interrupt selection register 4 (EXTI_EXTICR4)*.

Figure 140. EXTI mux GPIO selection



The EXTIs mux outputs are available as output signals from the EXTI to trigger other IPs. The EXTI mux outputs are available independently from any masking in EXTI_CmIMR1 and EXTI_CmEMR1.

23.4 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wakeup event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of a CPU(m) wakeup is conditioned by the CPU(m) interrupt mask and CPU(m) event mask.

Table 127. Masking functionality

CPU interrupt enable EXTI_CmIMR.IMn	CPU event enable EXTI_CmEMR.EMn	Configurable event inputs EXTI_RPR.RPIFn EXTI_FPR.FPIFn	exti(n) interrupt(1)	CPU(m) event	CPU(m) wakeup
0 for all CPUs	0	No	Masked	Masked	Masked
	1		Masked	Yes	Yes
1 for any CPU	0	Status latched	Yes	Masked	Yes(2)
	1		Yes	Yes	Yes

1. The single exti(n) interrupt goes to all CPUs. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU(m) interrupt controller.
2. Only if CPU(m) interrupt is enabled in EXTI_CmIMR.IMn.

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU(m) interrupt is unmasked the corresponding pending bit EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set and the CPU(m) sub-system is woken up and CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn pending bit must be cleared by software writing, it to ‘1’. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI.

When the associated CPU(m) interrupt is unmasked the corresponding CPU sub-system is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU(m) event must be unmasked to generate an event. When the enabled edge(s) occur on the event input a CPU(m) event pulse is generated. There is no event pending bit.

For the configurable event inputs an event request can be generated by software when writing a ‘1’ in the software interrupt/event register EXTI_SWIER, generating a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, irrespective of the setting in EXTI_RTSR.

23.5 EXTI TrustZone security

The EXTI is able to secure register bits from being modified by non-secure accesses. The TrustZone security can individually be activated per input event via the register bits in EXTI_TZENRn. At EXTI level the security consists in preventing a non-secure write access to:

- Change the settings of the secure configurable events.
- Change the masking of the secure input events.
- Clear pending status of the secure input events.

Note that the EXTI does not differentiate secure accesses from individual CPUs. Any CPU secure access can modify the secure register bits.

Table 128. Register security overview

Register name	Access type	Security(1)
EXTI_RTSRn	RW	Non-secure, security can be bit wise enabled in EXTI_TZENRn
EXTI_FTSRn	RW	
EXTI_SWIERn	RW	
EXTI_RPRn	RW	
EXTI_FPRn	RW	
EXTI_TZENRn	RW	Always secure
EXTI_EXTICRx	RW	Non-secure, security can be field wise enabled in EXTI_TZENRn
EXTI_CmIMRn	RW	Non-secure, security can be bit wise enabled in EXTI_TZENRn
EXTI_CmEMRn	RW	
EXTI_HWCFGRn	R	Non-secure
EXTI_VER	R	
EXTI_ID	R	
EXTI_SID	R	

1. Security is enabled with the individual Input event. EXTI_TZENRn registers.

When TZ security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access, a non-secure write access is discarded and a read returns 0.

When input events are non-secure, the TZ security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and non-secure access.

23.6 EXTI registers

The EXTI register map is divided as detailed in *Table 129*.

Table 129. EXTI register map sections

Address	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [63:32] configuration
0x040 - 0x05C	General configurable event [95:64] configuration
0x060 - 0x06C	EXTI IOport mux selection
0x080 - 0x0BC	CPU1 input event configuration
0x0C0 - 0x0FC	CPU2 input event configuration

All the registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

23.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RT[16:0]:** Rising trigger event configuration bit of configurable event input x (x = 16 to 0)(1).

When TZENx is disabled, RTx can be accessed with non-secure and secure access.

When TZENx is enabled, RTx can only be accessed with secure access. Non-secure write to RTx is discarded, non-secure read returns 0.

0: Rising trigger disabled (for event and Interrupt) for input line
1: Rising trigger enabled (for event and Interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

23.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FT[16:0]:** Falling trigger event configuration bit of configurable event input x (x = 16 to 0)(1).

When TZENx is disabled, FTx can be accessed with non-secure and secure access.

When TZENx is enabled, FTx can only be accessed with secure access. Non-secure write to FTx is discarded, non-secure read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs. If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

23.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **SWI[16:0]:** Software interrupt on event x (x = 16 to 0)

When TZENx is disabled, SWIx can be accessed with non-secure and secure access.

When TZENx is enabled, SWIx can only be accessed with secure access. Non-secure write to SWx is discarded.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event x.

This bit is auto cleared by HW.

23.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RPIF[16:0]:** configurable event inputs x (x = 16 to 0) rising edge Pending bit.

When TZENx is disabled, RPIFx can be accessed with non-secure and secure access.

When TZENx is enabled, RPIFx can only be accessed with secure access. Non-secure write to RPIFx is discarded, non-secure read returns 0.

0: No rising edge trigger request occurred
1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

23.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FPIF[**16:0**]:** configurable event inputs x (x = 16 to 0) falling edge pending bit.

When TZENx is disabled, FPIFx can be accessed with non-secure and secure access.

When TZENx is enabled, FPIFx can only be accessed with secure access. Non-secure write to FPIFx is discarded, non-secure read returns 0.

0: No falling edge trigger request occurred
1: rising edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

23.6.6 EXTI TrustZone enable register (EXTI_TZENR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides TrustZone Write access security, a non-secure write access will generate a bus error. A non-secure read will return the register data.

Contains only register bits for TrustZone capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TZEN 26	Res.	TZEN 24	Res.	Res.	Res.	Res.	TZEN 19	TZEN 18	TZEN 17	Res.
					rw		rw					rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZEN 15	TZEN 14	TZEN 13	TZEN 12	TZEN 11	TZEN 10	TZEN9	TZEN8	TZEN7	TZEN6	TZEN5	TZEN4	TZEN3	TZEN2	TZEN1	TZEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **TZEN26:** TrustZone security enable on event input 26.
- 0: Event TrustZone disabled (non-secure)
- 1: Event TrustZone enabled (secure)

Bit 25 Reserved, must be kept at reset value.

Bit 24 **TZEN24:** TrustZone security enable on event input 24.
- 0: Event TrustZone disabled (non-secure)
- 1: Event TrustZone enabled (secure)

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:17 **TZEN[19:17]:** TrustZone security enable on event input x (x = 19 to 17)
- 0: Event TrustZone disabled (non-secure)
- 1: Event TrustZone enabled (secure)

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **TZEN[15:0]:** TrustZone security enable on event input x (x = 15 to 0)
- 0: Event TrustZone disabled (non-secure)
- 1: Event TrustZone enabled (secure)

23.6.7 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.8 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.9 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.10 EXTI rising edge pending register (EXTI_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.11 EXTI falling edge pending register (EXTI_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.12 EXTI TrustZone enable register (EXTI_TZENR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides TrustZone Write access security, a non-secure write access will generate a bus error. A non-secure read will return the register data.

Contains only register bits for TrustZone capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TZEN60	TZEN59	TZEN58	TZEN57	TZEN56	TZEN55	TZEN54	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	TZEN41	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:22 **TZEN[60:54]:** TrustZone security enable on Event input x (x = 60 to 54)
- 0: Event TrustZone disabled (non-secure)
- 1: Event TrustZone enabled (secure)

Bits 21:10 Reserved, must be kept at reset value.

Bit 9 **TZEN41:** TrustZone security enable on Event input 41
- 0: Event TrustZone disabled (non-secure)
- 1: Event TrustZone enabled (secure)

Bits 8:0 Reserved, must be kept at reset value.

23.6.13 EXTI rising trigger selection register (EXTI_RTSR3)

Address offset: 0x040

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RT74	RT73	Res.	Res.	Res.	Res.	RT68	Res.	RT66	RT65	Res.
					rw	rw					rw		rw	rw	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RT74:** Rising trigger event configuration bit of configurable Event input 74(1)

When TZEN74 is disabled, RT74 can be accessed with non-secure and secure access.

When TZEN74 is enabled, RT74 can only be accessed with secure access. Non-secure write to RT74 is discarded, non-secure read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

Bit 9 **RT73:** Rising trigger event configuration bit of configurable Event input 73(1)

When TZEN73 is disabled, RT73 can be accessed with non-secure and secure access.

When TZEN73 is enabled, RT73 can only be accessed with secure access. Non-secure write to RT73 is discarded, non-secure read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **RT68:** Rising trigger event configuration bit of configurable event input 68(1)

When TZEN68 is disabled, RT68 can be accessed with non-secure and secure access.

When TZEN68 is enabled, RT68 can only be accessed with secure access. Non-secure write to RT68 is discarded, non-secure read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

Bit 3 Reserved, must be kept at reset value.

Bit 2 **RT66:** Rising trigger event configuration bit of configurable event input 66(1)

When TZEN66 is disabled, RT66 can be accessed with non-secure and secure access.

When TZEN66 is enabled, RT66 can only be accessed with secure access. Non-secure write to RT66 is discarded, non-secure read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 1 **RT65:** Rising trigger event configuration bit of configurable event input 65(1)

When TZEN65 is disabled, RT65 can be accessed with non-secure and secure access.

When TZEN65 is enabled, RT65 can only be accessed with secure access. Non-secure write to RT65 is discarded, non-secure read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable Event input. In this case, both edges generate a trigger.

23.6.14 EXTI falling trigger selection register (EXTI_FTSR3)

Address offset: 0x044

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	FT74	FT73	Res.	Res.	Res.	Res.	FT68	Res.	FT66	FT65	Res.
					rw	rw					rw		rw	rw	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **FT74:** Falling trigger event configuration bit of configurable event input 74(1)

When TZEN74 is disabled, FT74 can be accessed with non-secure and secure access.

When TZEN74 is enabled, FT74 can only be accessed with secure access. Non-secure write to FT74 is discarded, non-secure read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 9 **FT73:** Falling trigger event configuration bit of configurable event input 73(1)

When TZEN73 is disabled, FT73 can be accessed with non-secure and secure access.

When TZEN73 is enabled, FT73 can only be accessed with secure access. Non-secure write to FT73 is discarded, non-secure read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **FT68:** Falling trigger event configuration bit of configurable event input 68[(1)]

When TZEN68 is disabled, FT68 can be accessed with non-secure and secure access.

When TZEN68 is enabled, FT68 can only be accessed with secure access. Non-secure write to FT68 is discarded, non-secure read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FT66:** Falling trigger event configuration bit of configurable event input 66[(1)]

When TZEN66 is disabled, FT66 can be accessed with non-secure and secure access.

When TZEN66 is enabled, FT66 can only be accessed with secure access. Non-secure write to FT66 is discarded, non-secure read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 1 **FT65:** Falling trigger event configuration bit of configurable event input 65[(1)]

When TZEN65 is disabled, FT65 can be accessed with non-secure and secure access.

When TZEN65 is enabled, FT65 can only be accessed with secure access. Non-secure write to FT65 is discarded, non-secure read returns 0.

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 0 Reserved, must be kept at reset value.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit will not be set.
Rising and falling edge triggers can be set for the same configurable Event input. In this case, both edges generate a trigger.

23.6.15 EXTI software interrupt event register (EXTI_SWIER3)

Address offset: 0x048

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	SWI74	SWI73	Res.	Res.	Res.	Res.	SWI68	Res.	SWI66	SWI65	Res.
					rw	rw					rw		rw	rw	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SWI74:** Software interrupt on event 74

When TZEN74 is disabled, SWI74 can be accessed with non-secure and secure access.

When TZEN74 is enabled, SWI74 can only be accessed with secure access. Non-secure write to SWI74 is discarded.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 74. This bit is auto cleared by HW.

Bit 9 **SWI73:** Software interrupt on event 73

When TZEN73 is disabled, SWI73 can be accessed with non-secure and secure access.

When TZEN73 is enabled, SWI73 can only be accessed with secure access. Non-secure write to SWI73 is discarded.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 73. This bit is auto cleared by HW.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **SWI68:** Software interrupt on event 68

When TZEN68 is disabled, SWI68 can be accessed with non-secure and secure access.

When TZEN68 is enabled, SWI68 can only be accessed with secure access. Non-secure write to SWI68 is discarded.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 68. This bit is auto cleared by HW.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SWI66:** Software interrupt on event 66

When TZEN66 is disabled, SWI66 can be accessed with non-secure and secure access.

When TZEN66 is enabled, SWI66 can only be accessed with secure access. Non-secure write to SWI66 is discarded.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 66. This bit is auto cleared by HW.

Bit 1 **SWI65:** Software interrupt on event 65

When TZEN65 is disabled, SWI65 can be accessed with non-secure and secure access.

When TZEN65 is enabled, SWI65 can only be accessed with secure access. Non-secure write to SWI65 is discarded.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. Will always return 0 when read.

0: Writing 0 has no effect.

1: Writing a 1 to this bit will trigger a rising edge event on event 65. This bit is auto cleared by HW.

Bit 0 Reserved, must be kept at reset value.

23.6.16 EXTI rising edge pending register (EXTI_RPR3)

Address offset: 0x04C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RPIF74	RPIF73	Res.	Res.	Res.	Res.	RPIF68	Res.	RPIF66	RPIF65	Res.
					rc_w1	rc_w1					rc_w1		rc_w1	rc_w1	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RPIF74:** configurable event inputs 74 rising edge pending bit.

When TZEN74 is disabled, RPIF74 can be accessed with non-secure and secure access.

When TZEN74 is enabled, RPIF74 can only be accessed with secure access. Non-secure write to RPIF74 is discarded, non-secure read returns 0.

0: No rising edge trigger request occurred
1: rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 9 **RPIF73:** configurable event inputs 73 rising edge pending bit.

When TZEN73 is disabled, RPIF73 can be accessed with non-secure and secure access.

When TZEN73 is enabled, RPIF73 can only be accessed with secure access. Non-secure write to RPIF73 is discarded, non-secure read returns 0.

0: No rising edge trigger request occurred
1: rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **RPIF68:** configurable event inputs 68 rising edge pending bit.

When TZEN68 is disabled, RPIF68 can be accessed with non-secure and secure access.

When TZEN68 is enabled, RPIF68 can only be accessed with secure access. Non-secure write to RPIF68 is discarded, non-secure read returns 0.

0: No rising edge trigger request occurred
1: rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **RPIF66:** configurable event inputs 66 rising edge pending bit.

When TZEN66 is disabled, RPIF66 can be accessed with non-secure and secure access.

When TZEN66 is enabled, RPIF66 can only be accessed with secure access. Non-secure write to RPIF66 is discarded, non-secure read returns 0.

0: No rising edge trigger request occurred
1: rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 1 **RPIF65:** configurable event inputs 65 rising edge pending bit.

When TZEN65 is disabled, RPIF65 can be accessed with non-secure and secure access.

When TZEN65 is enabled, RPIF65 can only be accessed with secure access. Non-secure write to RPIF65 is discarded, non-secure read returns 0.

0: No rising edge trigger request occurred
1: rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 0 Reserved, must be kept at reset value.

23.6.17 EXTI falling edge pending register (EXTI_FPR3)

Address offset: 0x050

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	FPIF74	FPIF73	Res.	Res.	Res.	Res.	FPIF68	Res.	FPIF66	FPIF65	Res.
					rc_w1	rc_w1					rc_w1		rc_w1	rc_w1	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **FPIF74:** configurable event inputs 74 falling edge pending bit.

When TZEN74 is disabled, FPIF74 can be accessed with non-secure and secure access.

When TZEN74 is enabled, FPIF74 can only be accessed with secure access. Non-secure write to FPIF74 is discarded, non-secure read returns 0.

0: No falling edge trigger request occurred
1: falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 9 **FPIF73:** configurable event inputs 734 falling edge pending bit.

When TZEN73 is disabled, FPIF73 can be accessed with non-secure and secure access.

When TZEN73 is enabled, FPIF73 can only be accessed with secure access. Non-secure write toFPIF73 is discarded, non-secure read returns 0.

0: No falling edge trigger request occurred
1: falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **FPIF68:** configurable event inputs 68 falling edge pending bit.

When TZEN68 is disabled, FPIF68 can be accessed with non-secure and secure access.

When TZEN68 is enabled, FPIF68 can only be accessed with secure access. Non-secure write to FPIF68 is discarded, non-secure read returns 0.

0: No falling edge trigger request occurred
1: falling edge trigger request occurred

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FPIF66:** configurable event inputs 66 falling edge pending bit
When TZEN66 is disabled, FPIF66 can be accessed with non-secure and secure access.
When TZEN66 is enabled, FPIF66 can only be accessed with secure access. Non-secure write to FPIF66 is discarded, non-secure read returns 0.
0: No falling edge trigger request occurred
1: falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 1 **FPIF65:** configurable event inputs 65 falling edge pending bit.
When TZEN65 is disabled, FPIF65 can be accessed with non-secure and secure access.
When TZEN65 is enabled, FPIF65 can only be accessed with secure access. Non-secure write to FPIF65 is discarded, non-secure read returns 0.
0: No falling edge trigger request occurred
1: falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

Bit 0 Reserved, must be kept at reset value.

23.6.18 EXTI TrustZone enable register (EXTI_TZENR3)

Address offset: 0x054

Reset value: 0x0000 0000

This register provides TrustZone Write access security, a non-secure write access will generate a bus error. A non-secure read will return the register data.

Contains only register bits for TrustZone capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.19 EXTI external interrupt selection register 1 (EXTI_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the NBIOPORT[7:0] configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.

When TZEN3 is disabled, EXTI3 can be accessed with non-secure and secure access.

When TZEN3 is enabled, EXTI3 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[3] pin
0x01: PB[3] pin
0x02: PC[3] pin
0x03: PD[3] pin
0x04: PE[3] pin
0x05: PF[3] pin
0x06: PG[3] pin
0x07: PH[3] pin
0x08: PI[3] pin
0x09: PJ[3] pin
0x0A: PK[3] pin
0x0B: PZ[3] pin
Others: Reserved

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.

When TZEN2 is disabled, EXTI2 can be accessed with non-secure and secure access.

When TZEN2 is enabled, EXTI2 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[2] pin
0x01: PB[2] pin
0x02: PC[2] pin
0x03: PD[2] pin
0x04: PE[2] pin
0x05: PF[2] pin
0x06: PG[2] pin
0x07: PH[2] pin
0x08: PI[2] pin
0x09: PJ[2] pin
0x0A: PK[2] pin
0x0B: PZ[2] pin
Others: Reserved

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.

When TZEN1 is disabled, EXTI1 can be accessed with non-secure and secure access.

When TZEN1 is enabled, EXTI1 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[1] pin
0x01: PB[1] pin
0x02: PC[1] pin
0x03: PD[1] pin
0x04: PE[1] pin
0x05: PF[1] pin
0x06: PG[1] pin
0x07: PH[1] pin
0x08: PI[1] pin
0x09: PJ[1] pin
0x0A: PK[1] pin
0x0B: PZ[1] pin
Others: Reserved

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.

When TZEN0 is disabled, EXTI0 can be accessed with non-secure and secure access.

When TZEN0 is enabled, EXTI0 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[0] pin
0x01: PB[0] pin
0x02: PC[0] pin
0x03: PD[0] pin
0x04: PE[0] pin
0x05: PF[0] pin
0x06: PG[0] pin
0x07: PH[0] pin
0x08: PI[0] pin
0x09: PJ[0] pin
0x0A: PK[0] pin
0x0B: PZ[0] pin
Others: Reserved

23.6.20 EXTI external interrupt selection register 2 (EXTI_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the NBIOPORT[7:0] configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.

When TZEN7 is disabled, EXTI7 can be accessed with non-secure and secure access.

When TZEN7 is enabled, EXTI7 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[7] pin
0x01: PB[7] pin
0x02: PC[7] pin
0x03: PD[7] pin
0x04: PE[7] pin
0x05: PF[7] pin
0x06: PG[7] pin
0x07: PH[7] pin
0x08: PI[7] pin
0x09: PJ[7] pin
0x0A: PK[7] pin
0x0B: PZ[7] pin
Others: Reserved

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.

When TZEN6 is disabled, EXTI6 can be accessed with non-secure and secure access.

When TZEN6 is enabled, EXTI6 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[6] pin
0x01: PB[6] pin
0x02: PC[6] pin
0x03: PD[6] pin
0x04: PE[6] pin
0x05: PF[6] pin
0x06: PG[6] pin
0x07: PH[6] pin
0x08: PI[6] pin
0x09: PJ[6] pin
0x0A: PK[6] pin
0x0B: PZ[6] pin
Others: Reserved

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.

When TZEN5 is disabled, EXTI5 can be accessed with non-secure and secure access.

When TZEN5 is enabled, EXTI5 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[5] pin
0x01: PB[5] pin
0x02: PC[5] pin
0x03: PD[5] pin
0x04: PE[5] pin
0x05: PF[5] pin
0x06: PG[5] pin
0x07: PH[5] pin
0x08: PI[5] pin
0x09: PJ[5] pin
0x0A: PK[5] pin
0x0B: PZ[5] pin
Others: Reserved

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.

When TZEN4 is disabled, EXTI4 can be accessed with non-secure and secure access.

When TZEN4 is enabled, EXTI4 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[4] pin
0x01: PB[4] pin
0x02: PC[4] pin
0x03: PD[4] pin
0x04: PE[4] pin
0x05: PF[4] pin
0x06: PG[4] pin
0x07: PH[4] pin
0x08: PI[4] pin
0x09: PJ[4] pin
0x0A: PK[4] pin
0x0B: PZ[4] pin
Others: Reserved

23.6.21 EXTI external interrupt selection register 3 (EXTI_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the NBIOPORT[7:0] configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.

When TZEN11 is disabled, EXTI11 can be accessed with non-secure and secure access.

When TZEN11 is enabled, EXTI11 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[11] pin
0x01: PB[11] pin
0x02: PC[11] pin
0x03: PD[11] pin
0x04: PE[11] pin
0x05: PF[11] pin
0x06: PG[11] pin
0x07: PH[11] pin
0x08: PI[11] pin
0x09: PJ[11] pin
0x0A: PK[11] pin
0x0B: PZ[11] pin
Others: Reserved

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.

When TZEN10 is disabled, EXTI10 can be accessed with non-secure and secure access.

When TZEN10 is enabled, EXTI10 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[10] pin
0x01: PB[10] pin
0x02: PC[10] pin
0x03: PD[10] pin
0x04: PE[10] pin
0x05: PF[10] pin
0x06: PG[10] pin
0x07: PH[10] pin
0x08: PI[10] pin
0x09: PJ[10] pin
0x0A: PK[10] pin
0x0B: PZ[10] pin
Others: Reserved

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.

When TZEN9 is disabled, EXTI9 can be accessed with non-secure and secure access.

When TZEN9 is enabled, EXTI9 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[9] pin
0x01: PB[9] pin
0x02: PC[9] pin
0x03: PD[9] pin
0x04: PE[9] pin
0x05: PF[9] pin
0x06: PG[9] pin
0x07: PH[9] pin
0x08: PI[9] pin
0x09: PJ[9] pin
0x0A: PK[9] pin
0x0B: PZ[9] pin
Others: Reserved

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.

When TZEN8 is disabled, EXTI8 can be accessed with non-secure and secure access.

When TZEN8 is enabled, EXTI8 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[8] pin
0x01: PB[8] pin
0x02: PC[8] pin
0x03: PD[8] pin
0x04: PE[8] pin
0x05: PF[8] pin
0x06: PG[8] pin
0x07: PH[8] pin
0x08: PI[8] pin
0x09: PJ[8] pin
0x0A: PK[8] pin
0x0B: PZ[8] pin
Others: Reserved

23.6.22 EXTI external interrupt selection register 4 (EXTI_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

EXTIm fields contain only the number of bits in line with the NBIOPORT[7:0] configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.

When TZEN15 is disabled, EXTI15 can be accessed with non-secure and secure access.

When TZEN15 is enabled, EXTI15 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[15] pin
0x01: PB[15] pin
0x02: PC[15] pin
0x03: PD[15] pin
0x04: PE[15] pin
0x05: PF[15] pin
0x06: PG[15] pin
0x07: PH[15] pin
0x08: PI[15] pin
0x09: PJ[15] pin
0x0A: PK[15] pin
0x0B: PZ[15] pin
Others: Reserved

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.

When TZEN14 is disabled, EXTI14 can be accessed with non-secure and secure access.

When TZEN14 is enabled, EXTI14 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[14] pin
0x01: PB[14] pin
0x02: PC[14] pin
0x03: PD[14] pin
0x04: PE[14] pin
0x05: PF[14] pin
0x06: PG[14] pin
0x07: PH[14] pin
0x08: PI[14] pin
0x09: PJ[14] pin
0x0A: PK[14] pin
0x0B: PZ[14] pin
Others: Reserved

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.

When TZEN13 is disabled, EXTI13 can be accessed with non-secure and secure access.

When TZEN13 is enabled, EXTI13 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[13] pin
0x01: PB[13] pin
0x02: PC[13] pin
0x03: PD[13] pin
0x04: PE[13] pin
0x05: PF[13] pin
0x06: PG[13] pin
0x07: PH[13] pin
0x08: PI[13] pin
0x09: PJ[13] pin
0x0A: PK[13] pin
0x0B: PZ[13] pin
Others: Reserved

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.

When TZEN12 is disabled, EXTI12 can be accessed with non-secure and secure access.

When TZEN12 is enabled, EXTI12 can only be accessed with secure access. Non-secure write is discarded, non-secure read returns 0.

0x00: PA[12] pin
0x01: PB[12] pin
0x02: PC[12] pin
0x03: PD[12] pin
0x04: PE[12] pin
0x05: PF[12] pin
0x06: PG[12] pin
0x07: PH[12] pin
0x08: PI[12] pin
0x09: PJ[12] pin
0x0A: PK[12] pin
0x0B: PZ[12] pin
Others: Reserved

23.6.23 EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR1)

Address offset: 0x080

Reset value: 0xFFFE 0000

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IM[31:0]:** CPU wakeup with interrupt mask on event input x (x = 31:0)(1)(2).

When TZENx is disabled, IMx can be accessed with non-secure and secure access.

When TZENx is enabled, IMx can only be accessed with secure access. Non-secure write to IMx is discarded, non-secure read returns 0.

0: wakeup with interrupt request from Line x is masked
1: wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

23.6.24 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR1)

Address offset: 0x0C0

Reset value: 0xFFFE 0000

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IM[31:0]:** CPU2 wakeup with interrupt mask on event input x (x = 31 to 0)(1)(2).

When TZENx is disabled, IMx can be accessed with non-secure and secure access.

When TZENx is enabled, IMx can only be accessed with secure access. Non-secure write to IMx is discarded, non-secure read returns 0.

0: wakeup with interrupt request from Line x is masked
1: wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

23.6.25 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR1)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM19	EM18	EM17	Res.
												rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:17 **EM[19:17]:** CPU2 wakeup with event generation mask on event input x (x = 19 to 17)
When TZENx is disabled, EMx can be accessed with non-secure and secure access.
When TZENx is enabled, EMx can only be accessed with secure access. Non-secure write to EMx is discarded, non-secure read returns 0.
0: wakeup with event generation from Line x is masked
1: wakeup with event generation from Line x is unmasked

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EM[15:0]:** CPU2 wakeup with event generation mask on event input x (x = 15 to 0)
When TZENx is disabled, EMx can be accessed with non-secure and secure access.
When TZENx is enabled, EMx can only be accessed with secure access. Non-secure write to EMx is discarded, non-secure read returns 0.
0: wakeup with event generation from Line x is masked
1: wakeup with event generation from Line x is unmasked

23.6.26 EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR2)

Address offset: 0x090

Reset value: 0xFFFF FFFF

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM63	IM62	IM61	IM60	IM59	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	IM49	IM48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IM[63:32]:** CPU wakeup with interrupt mask on event input x (x = 63 to 32)(1)(2)
When TZENx is disabled, IMx can be accessed with non-secure and secure access.
When TZENx is enabled, IMx can only be accessed with secure access. Non-secure write to IMx is discarded, non-secure read returns 0.
0: wakeup with interrupt request from Line x is masked
1: wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

23.6.27 EXTI CPU2 wakeup with interrupt mask register (EXTI_C2IMR2)

Address offset: 0x0D0

Reset value: 0xFFFF FFFF

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM63	IM62	IM61	IM60	IM59	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	IM49	IM48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IM[63:32]:** CPU wakeup with interrupt mask on event input x (x = 63 to 32)(1)(2)

When TZENx is disabled, IMx can be accessed with non-secure and secure access.

When TZENx is enabled, IMx can only be accessed with secure access. Non-secure write to IMx is discarded, non-secure read returns 0.

0: wakeup with interrupt request from Line x is masked

1: wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
2. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

23.6.28 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR2)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.29 EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR3)

Address offset: 0x0A0

Reset value: 0x0000 09E9

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	IM75	IM74	IM73	IM72	IM71	IM70	IM69	IM68	IM67	IM66	IM65	IM64
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **IM[75:64]:** CPU wakeup with interrupt mask on event input x (x = 75 to 64)[(1)(2)]
When TZENx is disabled, IMx can be accessed with non-secure and secure access.
When TZENx is enabled, IMx can only be accessed with secure access. Non-secure write to IMx is discarded, non-secure read returns 0.
0: wakeup with interrupt request from Line x is masked
1: wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to '0' in order to disable the interrupt by default.
2. The reset value for Direct event inputs is set to '1' in order to enable the interrupt by default.

23.6.30 EXTI CPUm wakeup with interrupt mask register (EXTI_C2IMR3)

Address offset: 0x0E0

Reset value: 0x0000 0DE9

Contains register bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	IM75	IM74	IM73	IM72	IM71	IM70	IM69	IM68	IM67	IM66	IM65	IM64
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **IM[75:64]:** CPU2 wakeup with interrupt mask on event input x (x = 75 to 64)[(1)(2)]
When TZENx is disabled, IMx can be accessed with non-secure and secure access.
When TZENx is enabled, IMx can only be accessed with secure access. Non-secure write to IMx is discarded, non-secure read returns 0.
0: wakeup with interrupt request from Line x is masked
1: wakeup with interrupt request from Line x is unmasked

1. The reset value for configurable event inputs is set to '0' in order to disable the interrupt by default.
2. The reset value for Direct event inputs is set to '1' in order to enable the interrupt by default.

23.6.31 EXTI CPU2 wakeup with event mask register (EXTI_C2EMR3)

Address offset: 0x0E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM66	Res.	Res.
													rw		

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **EM66:** CPU2 wakeup with event mask on event input 66.
When TZEN66 is disabled, EM66 can be accessed with non-secure and secure access.
When TZEN66 is enabled, EM66 can only be accessed with secure access. Non-secure write to EM66 is discarded, non-secure read returns 0.
0: wakeup with event request from Line 66 is masked
1: wakeup with event request from Line 66 is unmasked

Bits 1:0 Reserved, must be kept at reset value.

23.6.32 EXTI hardware configuration register x (EXTI_HWCFGRx)

Address offset: 0x3C8 - 0x4 * (x-11), (x = 11 to 13)

Reset value: 0x050E FFFF (x = 11)

Reset value: 0x1FC0 2200 (x = 12)

Reset value: 0x0000 0000 (x = 13)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZ[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TZ[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TZ[31:0]:** HW configuration event TrustZone security capability.
0: Non-secure event.
1: TrustZone secure capable event.

23.6.33 EXTI hardware configuration register x (EXTI_HWCFGRx)

Address offset: 0x3D4 - 0x4 * (x - 8), (x = 8 to 10)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

23.6.34 EXTI hardware configuration register x (EXTI_HWCFGRx)

Address offset: 0x3E0 - 0x4 * (x - 5), (x = 5 to 7)

Reset value: 0x000E FFFF (x = 5)

Reset value: 0x0000 0000 (x = 6)

Reset value: 0x0000 0004 (x = 7)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CPUEVENT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPUEVENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CPUEVENT[31:0]:** HW configuration CPU event generation
0: No CPU event generated
1: CPU event generated

23.6.35 EXTI hardware configuration register x (EXTI_HWCFGRx)

Address offset: 0x3EC - 0x4 * (x - 2), (x = 2 to 4)

Reset value: 0x0001 FFFF (x = 2)

Reset value: 0x0000 0000 (x = 3)

Reset value: 0x0000 0616 (x = 4)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVENT_TRG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EVENT_TRG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EVENT_TRG[31:0]:** HW configuration event trigger type
0: Direct event
1: configurable event

23.6.36 EXTI hardware configuration register 1 (EXTI_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x000B 214B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBIOPORT[7:0]							
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPUEVTEN[3:0]				NBCPUS[3:0]				NBEVENTS[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **NBIOPORT[7:0]:** HW configuration of number of IO ports on EXTI (n+1).

Bits 15:12 **CPUEVTEN[3:0]:** HW configuration of CPU(m) event output enable.

Bits 11:8 **NBCPUS[3:0]:** HW configuration number of CPUs (n+1)

Bits 7:0 **NBEVENTS[7:0]:** HW configuration number of event (n+1)

23.6.37 EXTI IP version register (EXTI_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major Revision number

Bits 3:0 **MINREV[3:0]:** Minor Revision number

23.6.38 EXTI identification register (EXTI_IPIDR)

Address offset: 0x3F8

Reset value: 0x000E 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IPID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IPID[31:0]:** IP Identification

23.6.39 EXTI size ID register (EXTI_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

23.6.40 EXTI register map

The following table gives the EXTI register map and the reset values.

Table 130. EXTI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT[16:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT[16:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI[16:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF[16:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF[16:0]																
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	EXTI_TZENR1	Res.	Res.	Res.	Res.	Res.	TZEN26	Res.	TZEN24	Res.	Res.	Res.	Res.	TZEN[19:17]			Res.	TZEN[15:0]															
	Reset value						0		0					0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x020	EXTI_RTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x024	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x028	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x02C	EXTI_RPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x030	EXTI_FPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x034	EXTI_TZENR2	Res.	Res.	Res.	TZEN[60:54]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TZEN48	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0	0	0													0									
0x038-0x03C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x040	EXTI_RTSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT74	RT73	Res.	Res.	Res.	Res.	RT68	Res.	RT[66:65]		Res.
	Reset value																						0	0					0		0	0	

Table 130. EXTI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044	EXTI_FTSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT74	FT73	Res.	Res.	Res.	Res.	FT68	Res.	FT[66:65]		Res.
	Reset value																						0	0					0		0	0	
0x048	EXTI_SWIER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI74	SWI73	Res.	Res.	Res.	Res.	SWI68	Res.	SWI[66:65]		Res.
	Reset value																						0	0					0		0	0	
0x04C	EXTI_RPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF74	RPIF73	Res.	Res.	Res.	Res.	RPIF68	Res.	RPIF[66:65]		Res.
	Reset value																						0	0					0		0	0	
0x050	EXTI_FPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF74	FPIF73	Res.	Res.	Res.	Res.	FPIF68	Res.	FPIF[66:65]		Res.
	Reset value																						0	0					0		0	0	
0x054	EXTI_TZENR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TZEN66	Res.	
	Reset value																														0		
0x058-0x05C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x060	EXTI_EXTICR1	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	EXTI_EXTICR2	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	EXTI_EXTICR3	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	EXTI_EXTICR4	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI2[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070-0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080	EXTI_C1IMR1	IM[31:0].																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084-0x08C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x090	EXTI_C1IMR2	IM[63:32]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x094-0x09C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0A0	EXTI_C1IMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM[75:64]											
	Reset value																					1	0	0	1	1	1	1	0	1	0	0	1
0x0A4 - 0xBC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0C0	EXTI_C2IMR1	IM[31:0].																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 130. EXTI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C4	EXTI_C2EMR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM[19:17]			Res.	EM[15:0]															
	Reset value													0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0D0	EXTI_C2IMR2	IM[63:32]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0D4	EXTI_C2EMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0D8-0x0DC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x0E0	EXTI_C2IMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM[75:64]											
	Reset value																					1	1	0	1	1	1	1	0	1	0	0	1
0x0E4	EXTI_C2EMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM66	Res.	
	Reset value																														0		
0x0E8-0x3BC	Reserved	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C0	EXTI_HWCFGR13	TZ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C4	EXTI_HWCFGR12	TZ[31:0]																															
	Reset value	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
0x3C8	EXTI_HWCFGR11	TZ[31:0]																															
	Reset value	0	0	0	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3D4 - 0x4 * (x - 8), (x = 8 to 10)	EXTI_HWCFGRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x3D8	EXTI_HWCFGR7	CPUEVENT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x3DC	EXTI_HWCFGR6	CPUEVENT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3E0	EXTI_HWCFGR5	CPUEVENT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3E4	EXTI_HWCFGR4	EVENT_TRG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0
0x3E8	EXTI_HWCFGR3	EVENT_TRG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 130. EXTI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3EC	EXTI_HWCFGR2	EVENT_TRG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3F0	EXTI_HWCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBIOPORT[7:0]								CPUEVTEN[3:0]				NBCPUS[3:0]				NBEVENTS[7:0]							
	Reset value									0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1
0x3F4	EXTI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	1	0	0	0	0
0x3F8	EXTI_IPIDR	IPID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3FC	EXTI_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	0	1	1	1	0	1	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

24 Cyclic redundancy check calculation unit (CRC)

24.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

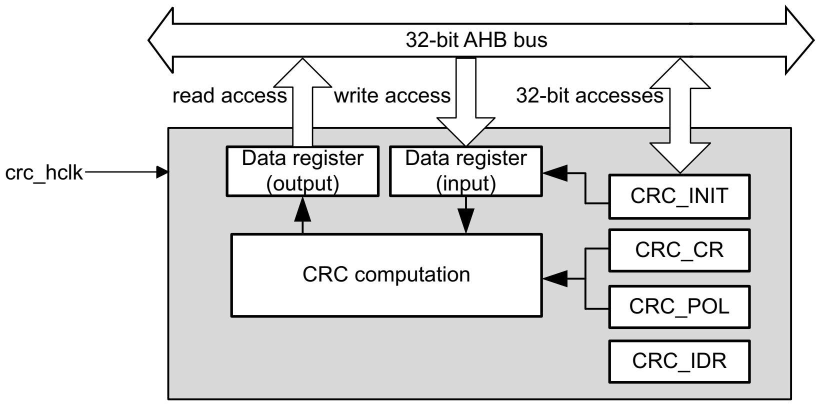
24.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

24.3 CRC functional description

24.3.1 CRC block diagram

Figure 141. CRC calculation unit block diagram



24.3.2 CRC internal signals

Table 131. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

24.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

24.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

24.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.
It holds the previous CRC calculation result when it is read.
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

24.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

24.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

This bitfield controls the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

24.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRC_INIT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC_INIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value
This register is used to write the CRC initial value.

24.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
POL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **POL[31:0]**: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

24.4.6 CRC hardware configuration register (CRC_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FULL_POLY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FULL_POLY[7:0]**: Polynomial type
00000000: Fixed polynomial 0x4C11DB7
00000001: Programmable polynomial coefficients and size (7,8,16,32)
Others: Reserved, must not be used.

24.4.7 CRC version register (CRC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits return the CRC major revision.

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits return the CRC minor revision.

24.4.8 CRC identification register (CRC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** Peripheral identifier
These bits return the CRC identifier:
ID[31:0] = 0x0017 0001

24.4.9 CRC size identification register (CRC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

These bits return the size of the memory region allocated to CRC registers. The size of this memory region is of 1 Kbyte:

SID[31:0] = 0xA3C5 DD01

24.4.10 CRC register map

Table 132. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1
0x3F0	CRC_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FULL_POLY[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x3F4	CRC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	0	0	0	1	1
0x3F8	CRC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 132. CRC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3FC	CRC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

25 Flexible memory controller (FMC)

The flexible memory controller (FMC) includes two memory controllers:

- the NOR/PSRAM memory controller
- the NAND memory controller

25.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, and NAND flash memory. Its main purposes are:

- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/One NAND flash memory
  - PSRAM (4 memory subregions)
  - Ferroelectric RAM (FRAM)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent chip-select control for each memory region
- Independent configuration for each memory region
- Write enable and byte lane select outputs for use with PSRAM, SRAM devices
- External asynchronous wait control

At startup the FMC pins must be configured by the user application. The FMC input/output pins which are not used by the application can be used for other purposes.

After reset, the FMC is disabled. Once all the used memory controllers are configured, the FMC must be enabled by setting the FMCEN bit.

The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. If some registers have to be re-configured, the FMC must be disabled (FMCEN bit set to 0x0). However, only few bits can be changed on the fly:

- FMCEN bit
- ECCEN and PBEN bits in FMC_PCR register
- FMC_CSQICR and FMC_CSQIER register bits
- FMC command sequencer registers and FMC BCH registers

- WEN bit in the FMC_PCR register

If some parameters have to be modified while the FMC is enabled, the FMC must be first disabled so that no further access is allowed to any memory controller during register modification. Once all needed configurations are updated, enable the FMC as follows:

1. Disable FMC by resetting the FMCEN bit.
2. Wait ISOST[1:0] bits in FMC_SR register to be set to 0b11.
3. Wait until PEF flag is set in the FMC_SR register.
4. Program the registers with new values.
5. Enable the FMC by setting the FMCEN bit.

25.2 FMC implementation

Table 133. FMC features

Modes/features	FMC
NAND flash Chip Select outputs	2
NAND flash Ready/Busy	1
NAND/NOR flash with common WAIT pin	Yes (NWAIT)

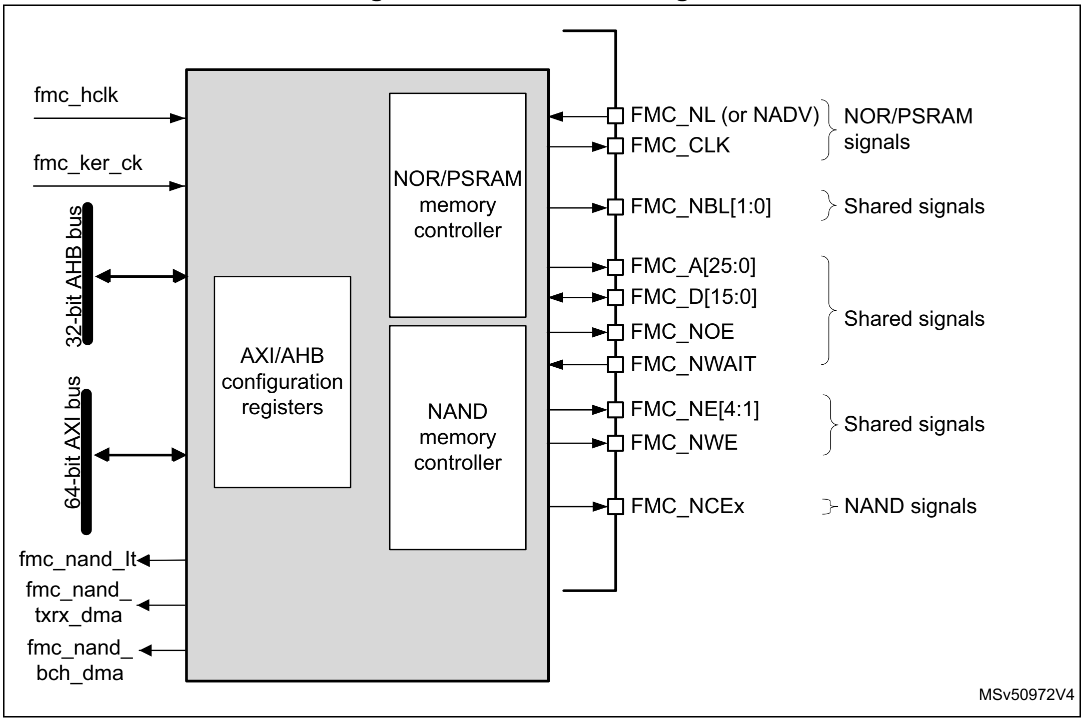
25.3 FMC block diagram

The FMC consists of the following main blocks:

- The NOR flash/PSRAM/SRAM controller
- The NAND controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the below figure.

Figure 142. FMC block diagram



1. The FMC features two NCEx chip select outputs.

25.4 FMC internal signals

Table 134 gives the list of FMC internal signals. FMC pins (or external signals) are described in *Section 25.8.1: External memory interface signals*.

Table 134. FMC internal signals

Names	Signal type	Description
fmc_nand_it	Digital output	FMC NAND flash interrupt
fmc_ker_ck	Digital input	FMC kernel clock
fmc_hclk	Digital input	FMC interface clock
fmc_nand_txrx_dma	Digital output	DMA request for reading or writing data from/to NAND flash memory
fmc_nand_bch_dma	Digital output	DMA request for BCH results

25.5 AHB interface

The AHB slave interface allows internal CPUs to configure the FMC registers.

The AHB clock (HCLK) is the reference clock for the FMC register accesses.

The FMC registers must be accessed by words (32 bits). For non 32-bit AHB accesses, FMC does not return an AHB bus error and the register value can be corrupted.

25.6 AXI interface

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32- or 16- or 8-bit accesses according to data size accesses.

The FMC generates an AXI slave error when one of the following conditions is met:

- When reading or writing to an FMC memory region (memory region 1 to 4) which is not enabled
- When reading or writing to any FMC memory region while the FMC is disabled (FMCEN bit is reset)
- When reading or writing by the command sequencer to NAND memory region while it is disabled
- When reading or writing to the NOR flash memory region while the MBKEN bit is reset in the FMC_BCRx register

The kernel clock for the FMC is the asynchronous fmc_ker_ck clock (refer the Reset and Clock control (RCC) section for details on fmc_ker_ck clock source selection).

25.6.1 Supported memories and transactions

General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AXI transaction size and memory data size must be equal to prevent issues from occurring.
- AXI transaction size is greater than the memory size:

  In this case, the FMC splits the AXI transaction into smaller consecutive memory accesses to meet the external data width.
- AXI transaction size is smaller than the memory size:

  The transfer may or not be consistent depending on the type of external device:
  - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM)

    In this case, the FMC allows read/write transactions and accesses the right data through its byte lanes NBL[1:0].

    Bytes to be written are addressed by NBL[1:0].

    All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.
  - Accesses to devices that do not have the byte select feature (NOR and NAND flash memories)

    This situation occurs when a byte access is requested to a 16-bit wide flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the flash memory), write transactions are not allowed while read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR flash/PSRAM

Synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:

- The read operation is split into two linear burst transactions.
- The write operation is split into two linear burst transactions.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 25.8.6*, for a detailed description of the NOR flash/PSRAM controller registers. Refer to *Section 25.9.9*, for a detailed description of the NAND flash registers.

25.7 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size memory regions of 256 Mbytes each (see *Table 135*):

- Memory region 1 is used to address up to 4 NOR flash memories or PSRAM devices. This region is split into four NOR/PSRAM subregions with four dedicated Chip Selects, as follows:
  - Subregion 1 - NOR/PSRAM 1
  - Subregion 2 - NOR/PSRAM 2
  - Subregion 3- NOR/PSRAM 3
  - Subregion 4 - NOR/PSRAM 4

Note: The Memory region 1 type attribute must be declared as 'device'.

- Memory region 3 is used to address NAND flash memory devices. The MPU memory attribute for this space must be reconfigured by software to device.

For each memory region, the type of memory to be used can be configured by the user application through the Configuration register.

Table 135. FMC memory regions (default mapping)

Address	Memory region	Supported memory type
0x6000 0000 - 0x6FFF FFFF	Memory region 1	NOR flash memory/PSRAM/SRAM
0x8000 0000 - 0x8FFF FFFF	Memory region 3	NAND flash memory

25.7.1 NOR/PSRAM address mapping

ADDR[27:26] bits are used to select one of the four memory subregions as shown in *Table 136*.

Table 136. NOR/PSRAM subregion selection

ADDR[27:26][(1)]	Selected subregion
00	Subregion 1 - NOR/PSRAM 1
01	Subregion 2 - NOR/PSRAM 2
10	Subregion 3- NOR/PSRAM 3
11	Subregion 4- NOR/PSRAM 4

1. ADDR are internal address lines that are translated to external memory.

The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 137. NOR/PSRAM External memory address

Memory width[(1)]	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	ADDR[25:0]	64 Mbytes * 8 = 512 Mbit
16-bit	ADDR[25:1] >> 1	64 Mbytes/2 * 16 = 512 Mbit

1. In case of a 16-bit external memory width, the FMC internally uses ADDR[25:1] to generate the address for external memory FMC_A[24:0].
Whatever the external memory width, FMC_A[0] must be connected to external memory address A[0].

25.7.2 NAND flash memory address mapping

The NAND memory region is divided into memory areas as indicated in *Table 138*.

Table 138. NAND memory mapping and timing registers

Start address	End address	FMC memory region	Memory space	Timing register
0x8800 0000	0x8FFF FFFF	Memory region 3 - NAND flash	Attribute	FMC_PATT (0x8C)
0x8000 0000	0x87FF FFFF		Common	FMC_PMEM (0x88)

For NAND flash memory, the common and attribute memory spaces are subdivided into three sections (see in *Table 139* below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 139. NAND flash memory region selection

Section name	ADDR[17:16]	Address range
Address section	10	0x20000-0x3FFFF
Command section	01	0x10000-0x1FFFF
Data section	00	0x00000-0x0FFFF

The ADDR[25:24] bits allow the selection of the NAND flash chip enable issued to the NAND memory as shown in *Table 140*. It allows switching from one chip enable to another in order to address a given NAND flash memory within a multiple-die package.

Table 140. NAND chip enable selection

ADDR[25:24]	Chip Select Selection
00	NCE1
01	NCE2
Other configuration	Reserved

The application software uses the 3 sections to access the NAND flash memory:

- **To send a command to NAND flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

Note: The NAND region has to be accessed as a device and not as memory. Write accesses to command and address sections must be done in Byte mode and aligned with the section base address (offset = 0).

Read and Write accesses to NAND data section must be in Byte mode for 8-bit NAND flash memories and Byte or Half-word mode for 16-bit NAND flash memories.

Read and Write bursts must start at section base address (offset = 0) and must not cross the 16-byte section boundary.

25.8 NOR flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per memory region. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait-states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a submultiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, memory region 1 must be configured in synchronous mode (see *Section 25.8.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID[1:0]) otherwise the FMC_CLK frequency is changed depending on AXI data transaction (refer to *Section 25.8.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each memory region is fixed and equal to 64 Mbytes. Each memory region is configured through dedicated registers (see *Section 25.8.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 141*) and support for wait management (for PSRAM and NOR flash accessed in burst mode).

Table 141. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	FMC clock cycle (fmc_ker_ck)	1	15
NBL setup	Duration of the Byte lanes setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	3
Data setup	Duration of the data setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	1	256
Data hold	Duration of the data hold phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	3
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	FMC clock cycle (fmc_ker_ck)	0	15
Clock divide ratio	Number of FMC clock cycles (fmc_ker_ck) to build one memory clock cycle (CLK)	Synchronous	FMC clock cycle (fmc_ker_ck)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (fmc_ker_ck)	2	17

25.8.1 External memory interface signals

Table 142, *Table 143* and *Table 144* list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals which are active low.

NOR flash memory, non-multiplexed I/Os

Table 142. Non-multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip Select, x = 1 to 4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR flash memory, 16-bit multiplexed I/Os

Table 143. 16-bit multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1 to 4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/SRAM/FRAM, non-multiplexed I/Os

Table 144. Non-multiplexed I/Os PSRAM/SRAM

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip Select, x = 1 to 4 (called NCE by PSRAM (CellularRAM™, CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 145. 16-bit multiplexed I/O PSRAM

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus

Table 145. 16-bit multiplexed I/O PSRAM (continued)

FMC signal name	I/O	Function
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1 to 4 (called NCE by PSRAM (CellularRAM™, CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

25.8.2 Supported memories and transactions

Table 146 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 146. NOR flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NOR flash (muxed I/Os and non-muxed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-
	Synchronous	R	64	16	Y	Split into 4 FMC accesses

Table 146. NOR flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32/64	16	Y	-
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32/64	16	Y	-
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	-
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses

25.8.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the fmc_ker_ck clock
- In synchronous mode (read or write), all output signals change on the rising edge of fmc_ker_ck clock.
- NBL signals

  They are applicable to memories which support byte enable.
  - The NBL signals are driven low:

    from the start of a read memory transfer and optionally earlier according to NBLSET, when the device requires some setup time from NBL to NCE,

    for bytes to be written during data transfer and according to the first data bytes during the address transfer.
  - The NBL signals are driven high two FMC_CLK clock cycles after the completion of the read or write transfer.

25.8.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR flash, PSRAM, SRAM, FRAM)

- Signals are synchronized by the internal clock fmc_ker_ck. This clock is not issued to the memory.
- The FMC always samples the data before deasserting the Chip Select signal NE. This guarantees that the memory data hold timing constraint is met (minimum chip enable high to data transition is usually 0 ns).
- If the extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B. The extended mode is not compatible with the muxed mode (EXTMOD bit must be kept cleared in muxed mode).
- If the extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in mode 1 or mode 2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP[1:0] = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP[1:0] = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 143. Mode 1 read access waveforms

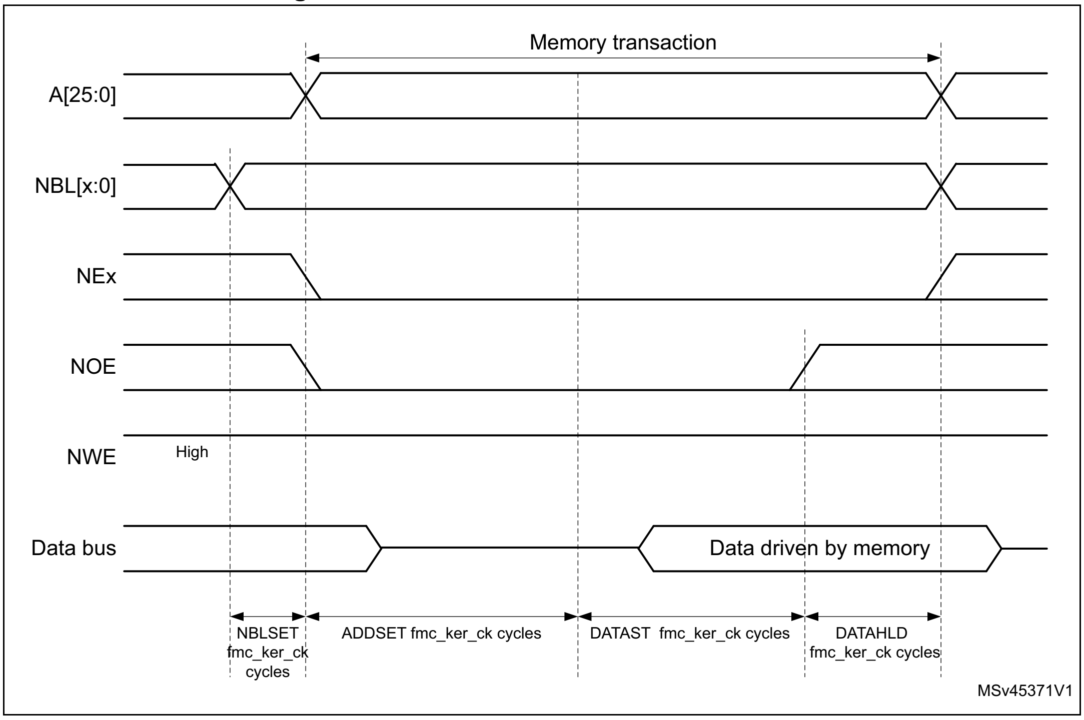
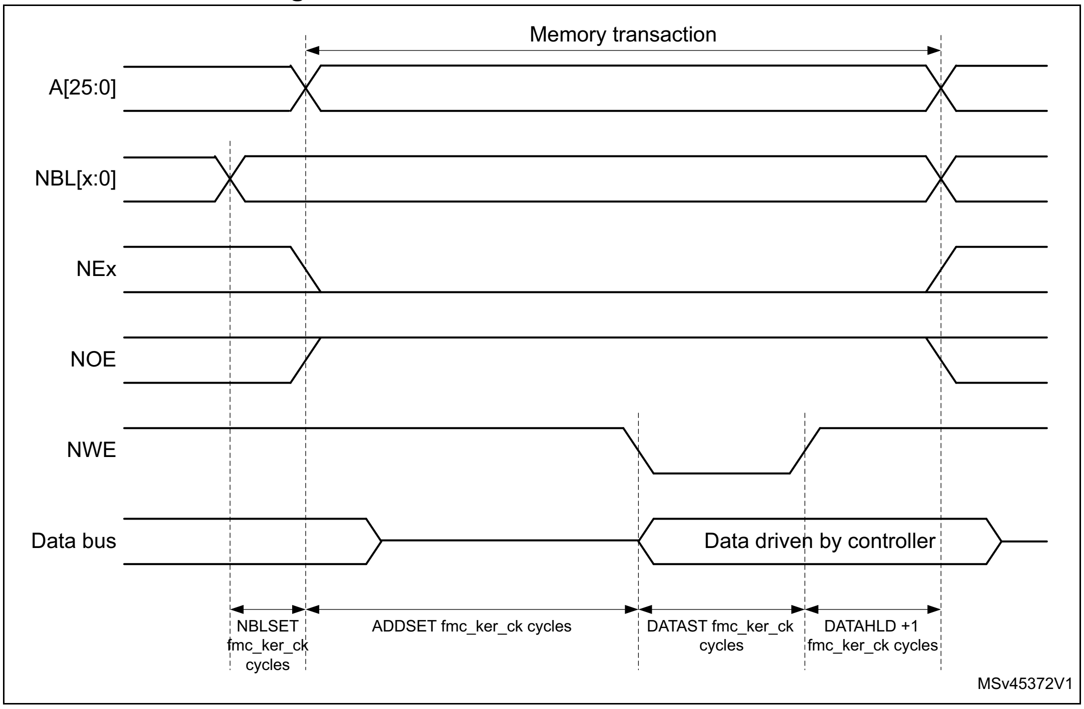


Figure 144. Mode 1 write access waveforms



The DATAHLD time at the end of the read and write transaction guarantee the address and data hold time after the NOE/NWE rising edge. For write, the DATAST value must be greater than zero (DATAST > 0).

Table 147. FMC_BCRx bitfields (mode 1)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5-4	MWID[1:0]	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 148. FMC_BTRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles) for read accesses.
29-28	ACCMOD	Don’t care
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles).

Table 148. FMC_BTRx bitfields (mode 1) (continued)

Bit number	Bit name	Value to set
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 0.

Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 145. Mode A read access waveforms

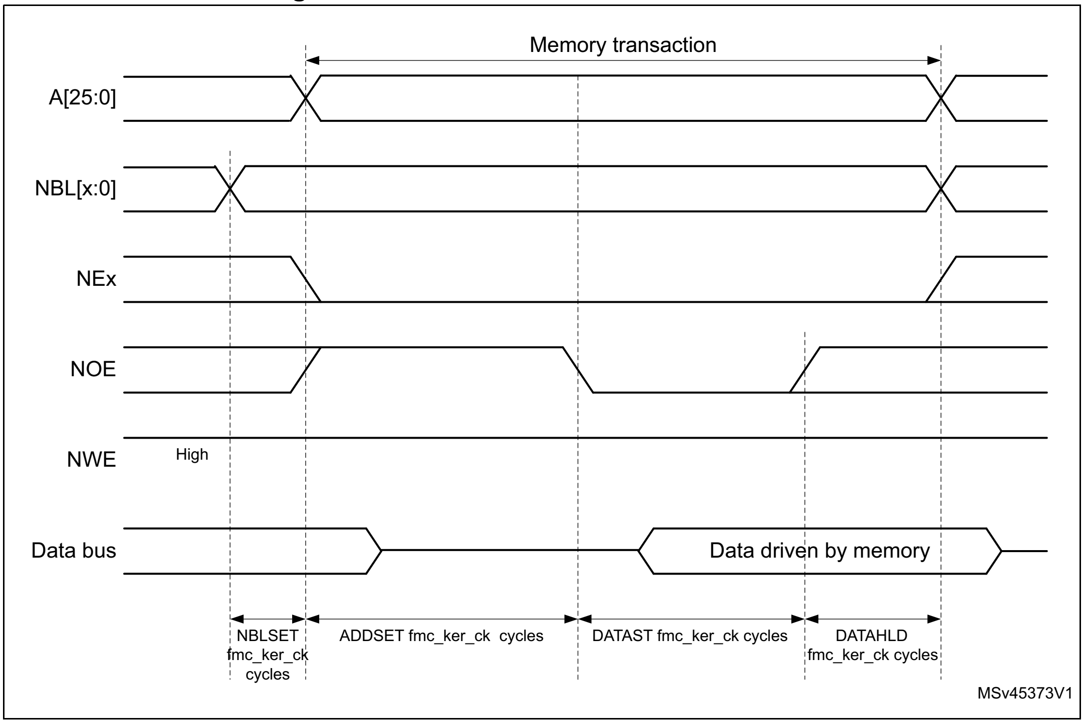
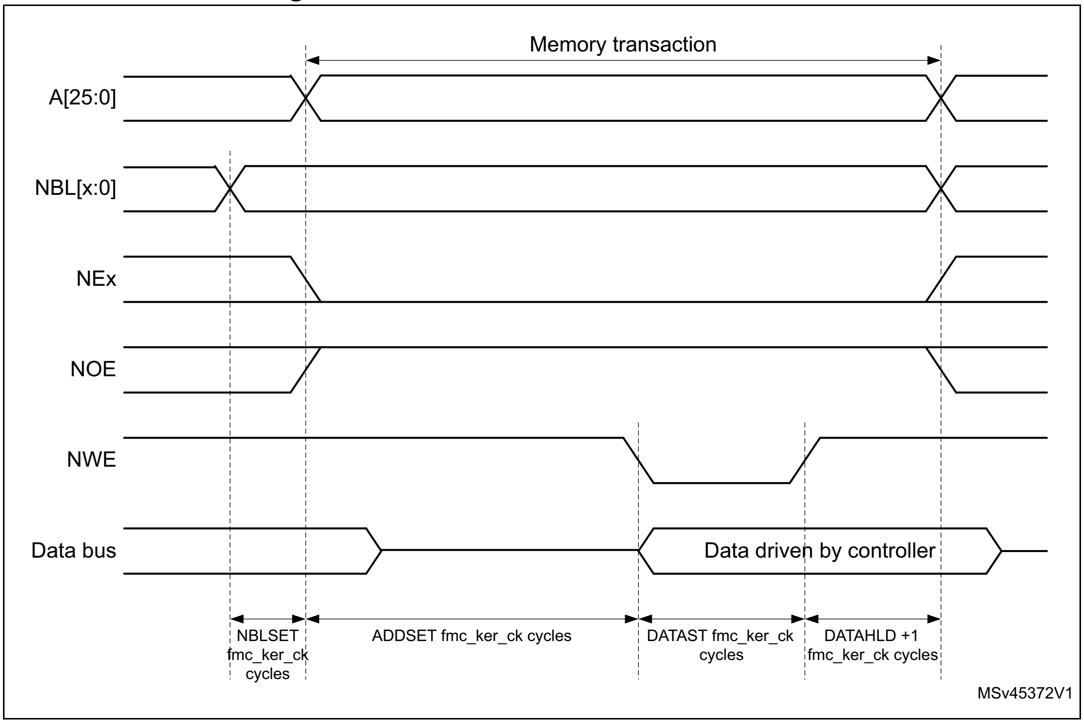


Figure 146. Mode A write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 149. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1

Table 149. FMC_BCRx bitfields (mode A) (continued)

Bit number	Bit name	Value to set
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don't care
5-4	MWID[1:0]	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 150. FMC_BTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x0
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 151. FMC_BWTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x0
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.

Table 151. FMC_BWTRx bitfields (mode A) (continued)

Bit number	Bit name	Value to set
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR flash

Figure 147. Mode 2 and mode B read access waveforms

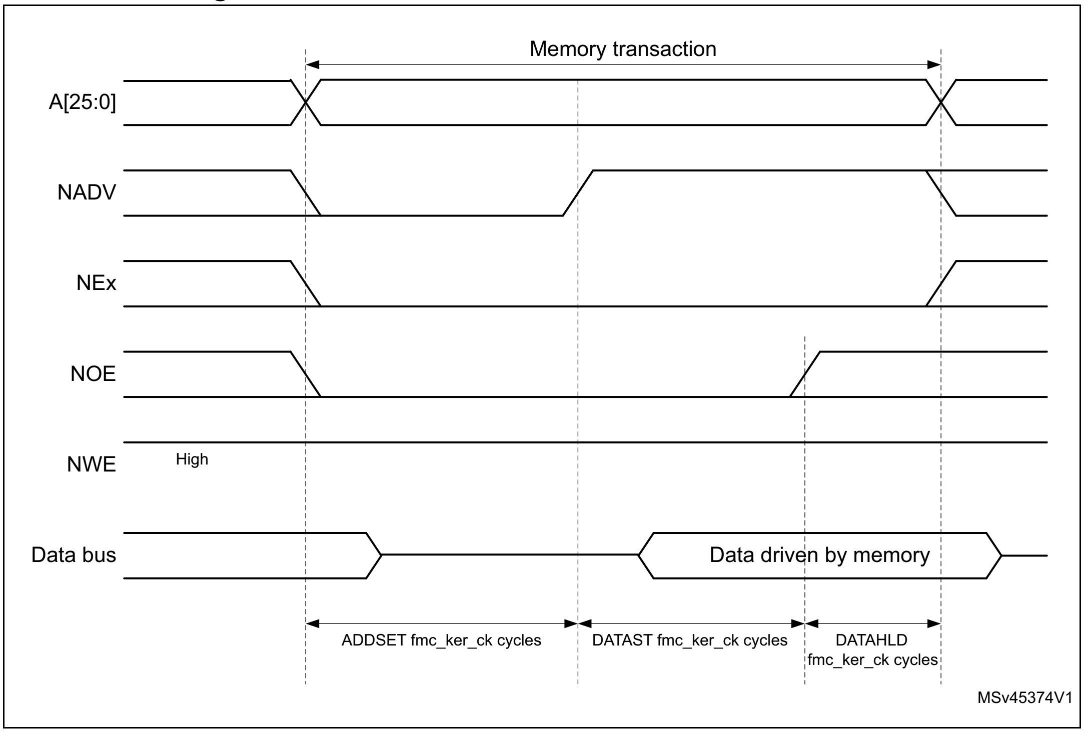


Figure 148. Mode 2 write access waveforms

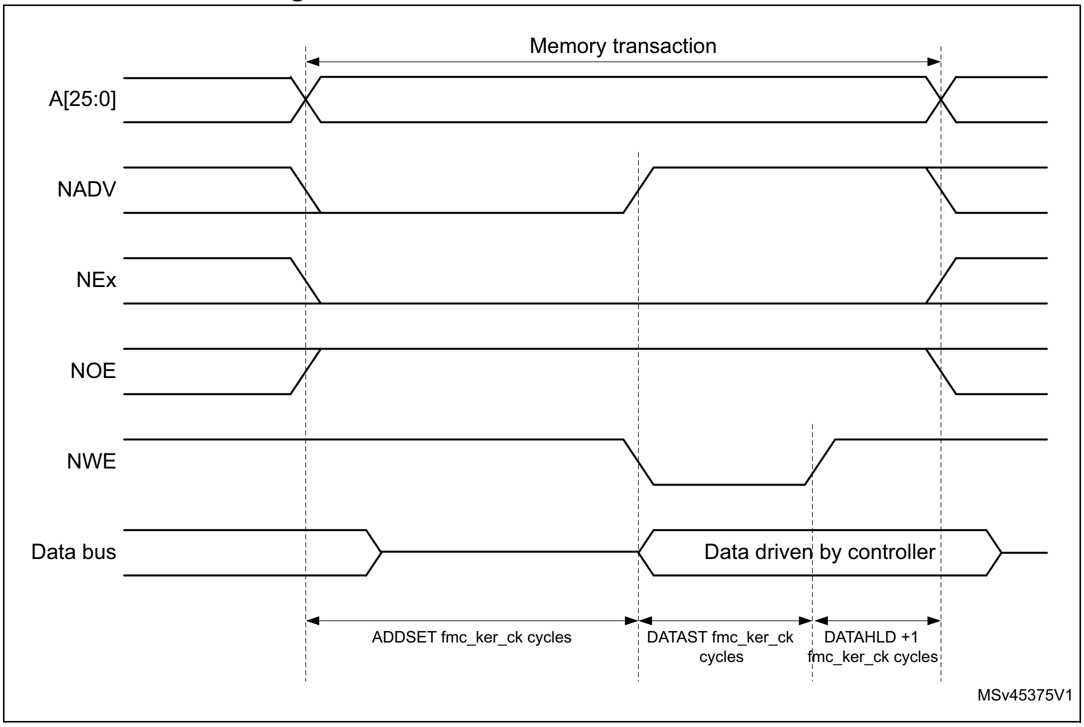
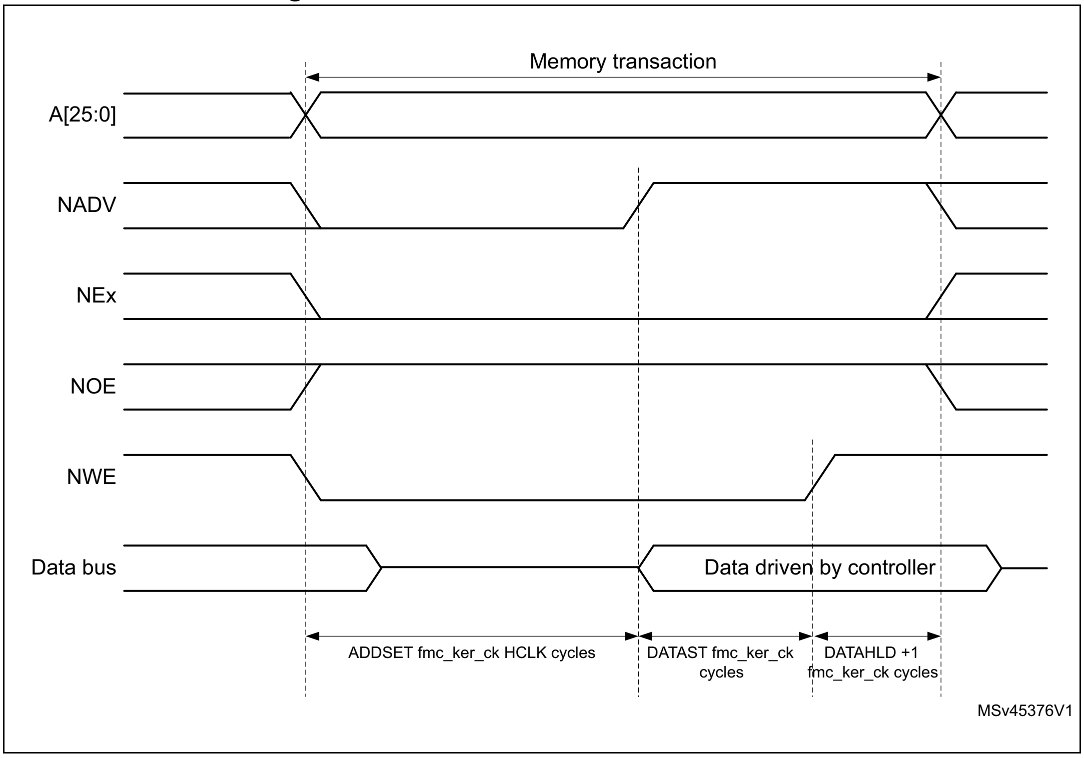


Figure 149. Mode B write access waveforms



The differences with mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 152. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID[1:0]	As needed
3-2	MTYP	0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 153. FMC_BTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x1 if extended mode is set
27-24	DATLAT	Don't care
23-20	CLKDIV	Don't care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.

Table 153. FMC_BTRx bitfields (mode 2/B) (continued)

Bit number	Bit name	Value to set
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 154. FMC_BWTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x1 if extended mode is set
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the extended mode is set (mode B), otherwise its content is don’t care.

Mode C - NOR flash - OE toggling

Figure 150. Mode C read access waveforms

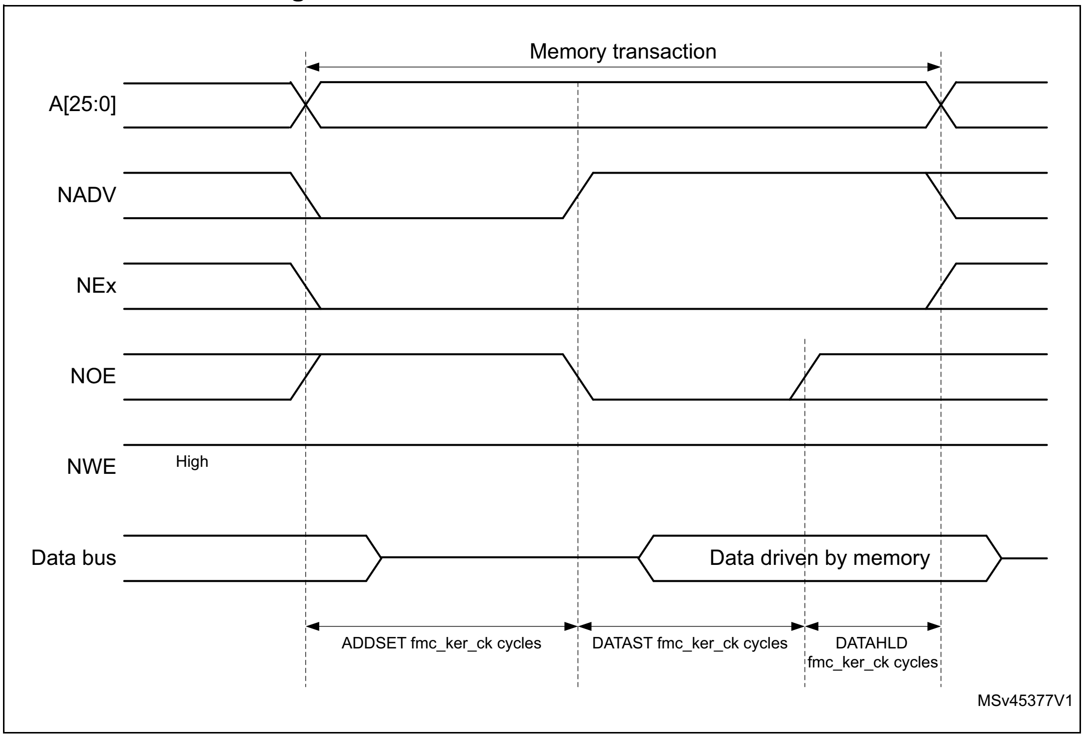
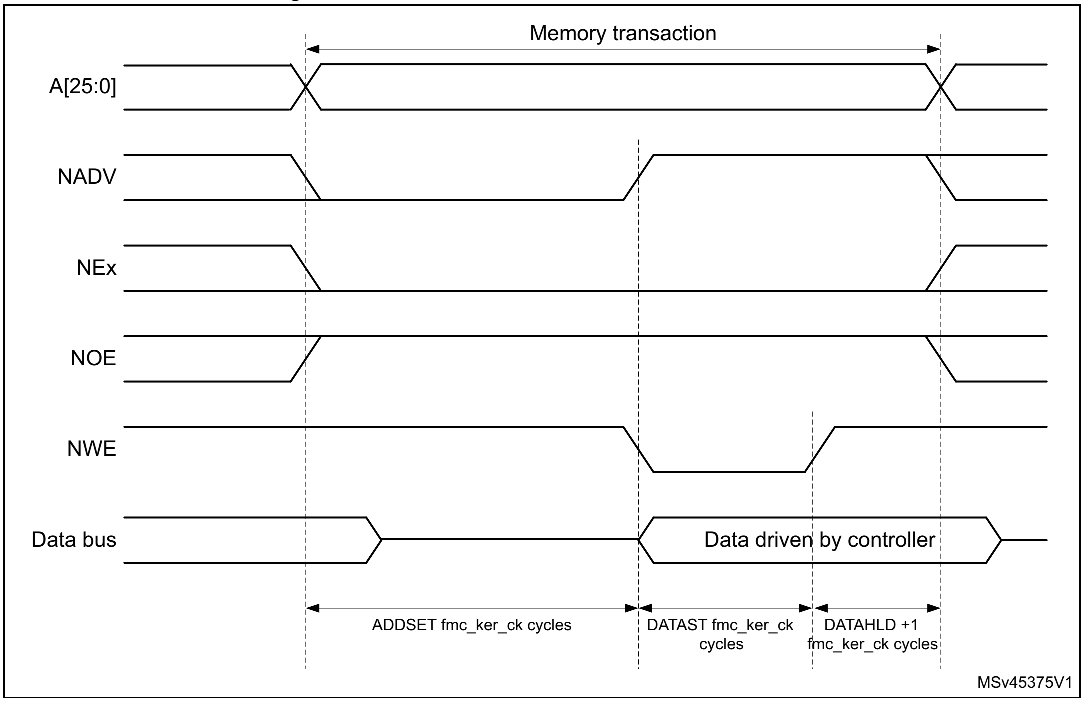


Figure 151. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 155. FMC_BCRx bitfields (mode C)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x02 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 156. FMC_BTRx bitfields (mode C)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 157. FMC_BWTRx bitfields (mode C)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x2
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 152. Mode D read access waveforms

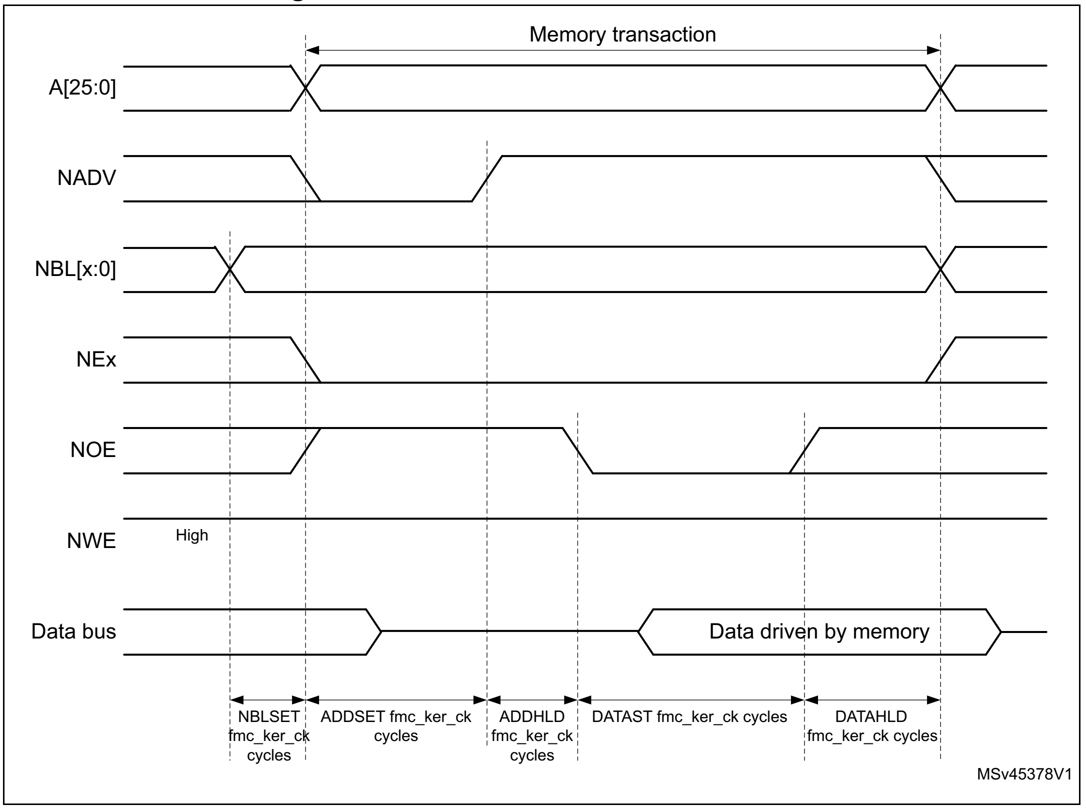
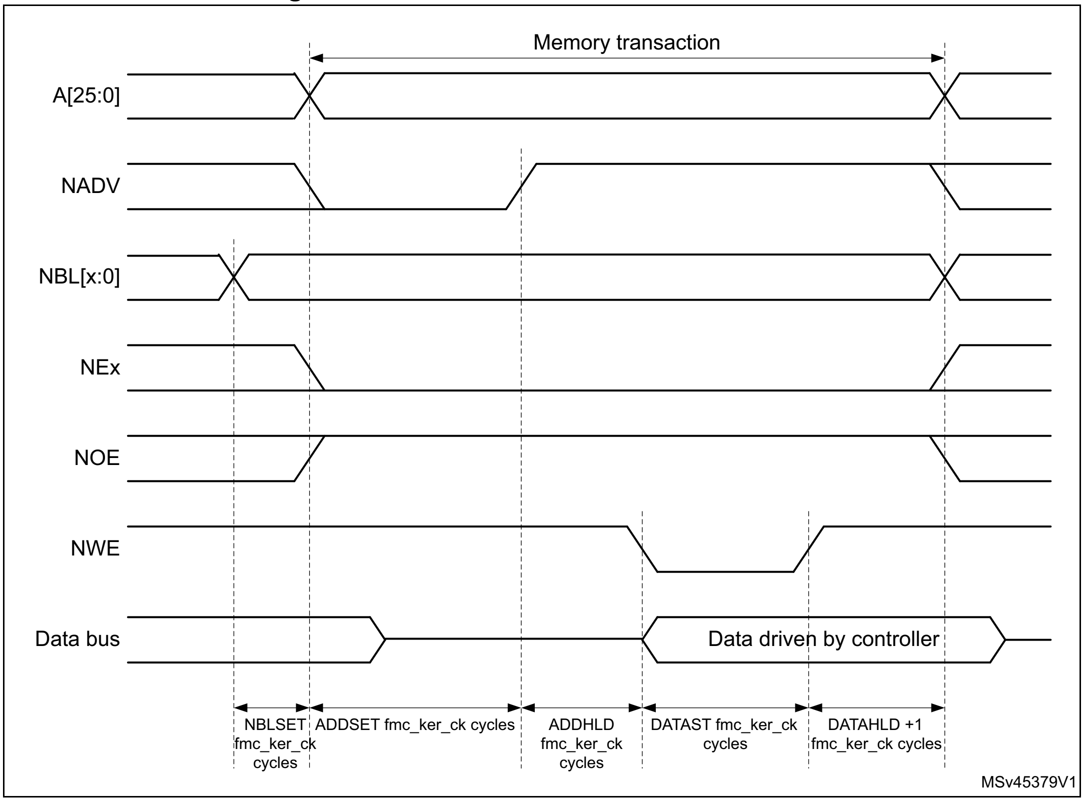


Figure 153. Mode D write access waveforms



The differences with mode 1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 158. FMC_BCRx bitfields (mode D)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared. The ASYNCWAIT feature is not supported in mode D mode.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 158. FMC_BCRx bitfields (mode D) (continued)

Bit No.	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 159. FMC_BTRx bitfields (mode D)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x3
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.

Table 160. FMC_BWTRx bitfields (mode D)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x3
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR flash memory

Figure 154. Muxed read access waveforms

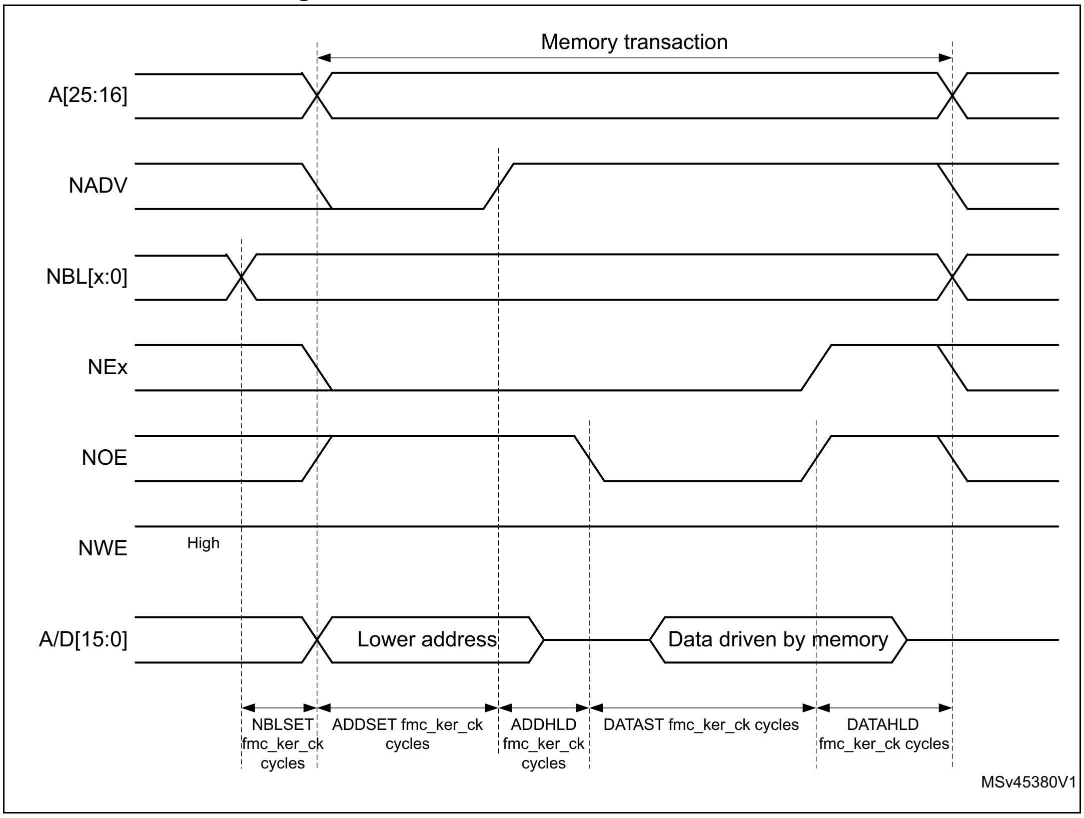
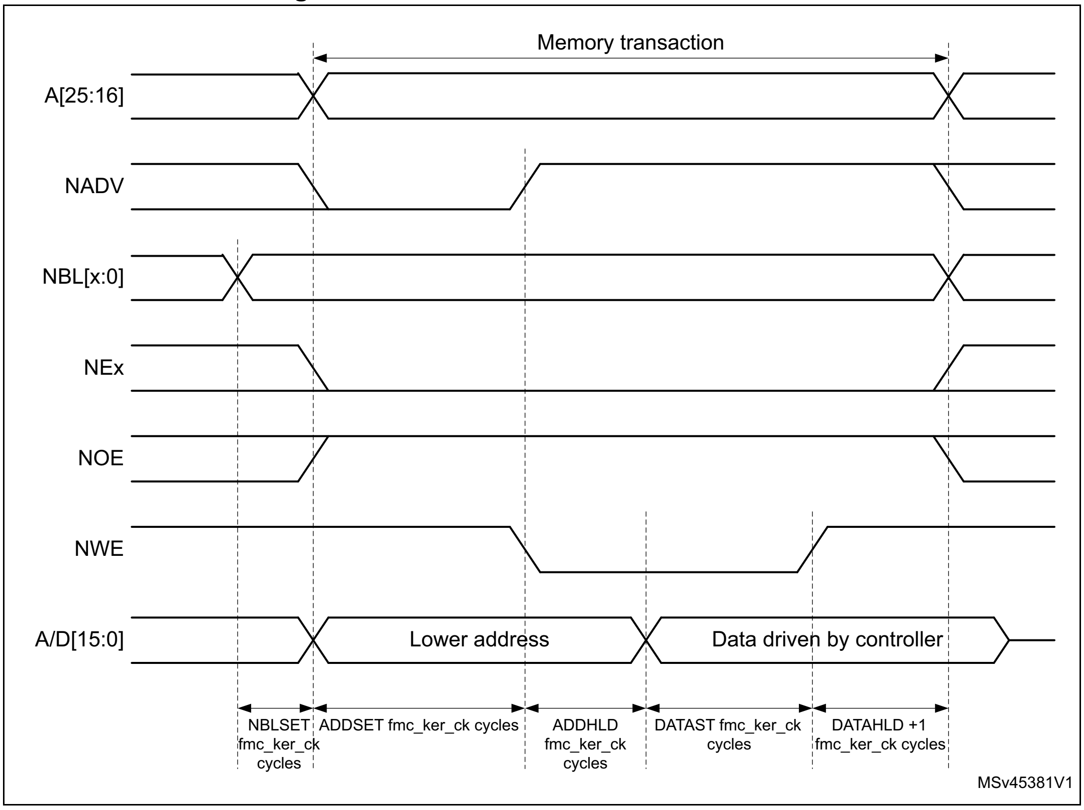


Figure 155. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 161. FMC_BCRx bitfields (muxed mode)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared. The ASYNCWAIT feature is not supported in muxed mode.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0

Table 161. FMC_BCRx bitfields (muxed mode) (continued)

Bit No.	Bit name	Value to set
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x2 (NOR flash memory)
1	MUXEN	0x1
0	MBKEN	0x1

Table 162. FMC_BTRx bitfields (muxed mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses, DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD fmc_ker_ck cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 1.

WAIT management for asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 fmc_ker_ck cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$\text{DATAST} \geq (4 \times \text{FMCCLK}) + \text{max\_wait\_assertion\_time}$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$\text{max\_wait\_assertion\_time} > \text{address\_phase} + \text{hold\_phase}$$

   then:

$$\text{DATAST} \geq (4 \times \text{FMCCLK}) + (\text{max\_wait\_assertion\_time} - \text{address\_phase} - \text{hold\_phase}$$

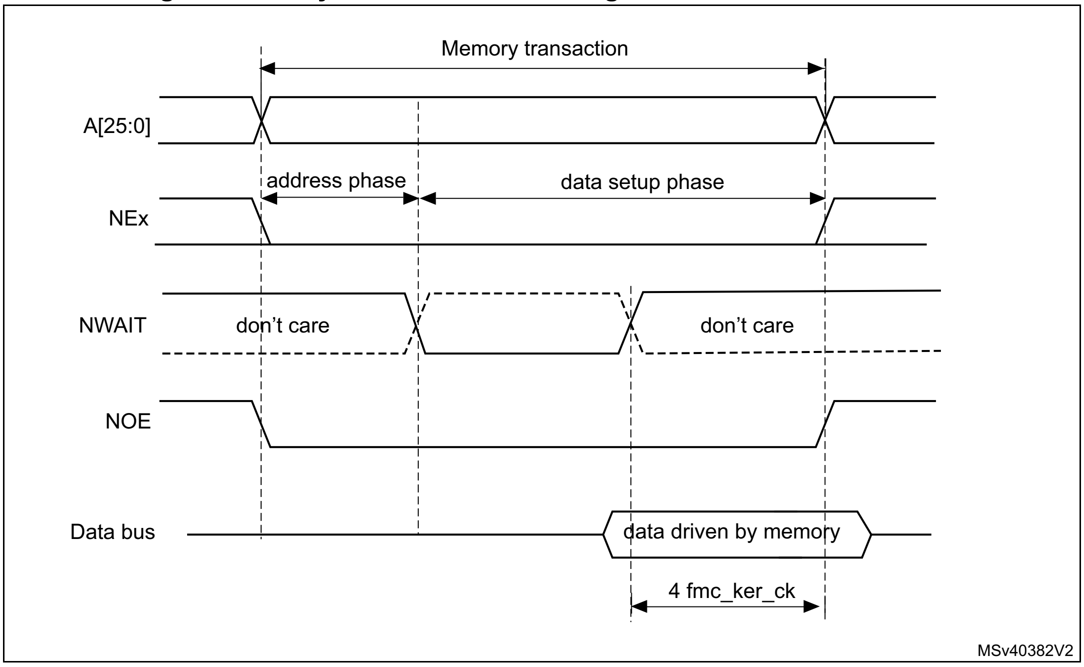
   otherwise

$$\text{DATAST} \geq 4 \times \text{FMCCLK}$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

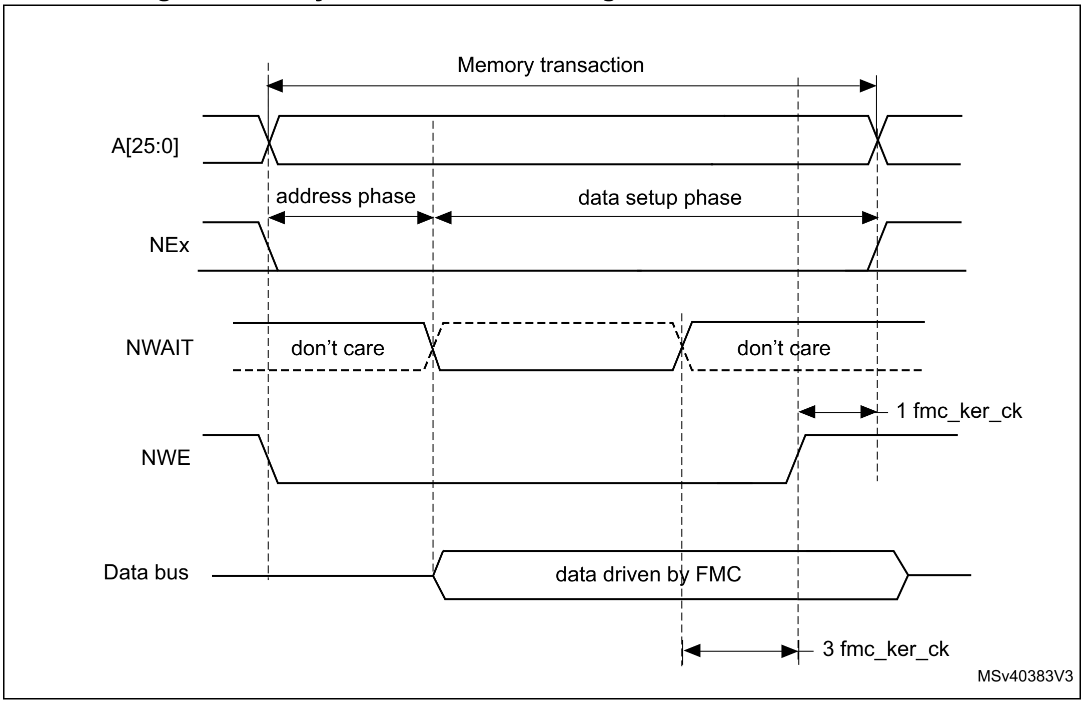
Figure 156 and *Figure 157* show the number of fmc_ker_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 156. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 157. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

CellularRAM™ (PSRAM) refresh management

The CellularRAM™ does not allow maintaining the chip select signal (NE) low for longer than the t_{CEM} timing specified in memory device. This timing can be programmed in the FMC_PCSCNTR register. It defines the maximum duration of the NE low pulse (expressed in fmc_ker_ck cycles) in synchronous mode for PSRAM read or write accesses.

25.8.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of fmc_ker_ck. It depends on the value of CLKDIV.

NOR flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:

- NOR flash latency = (DATLAT + 2) CLK clock cycles
- or NOR flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected memory region is configured in burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and deassert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Cross boundary page for CellularRAM™ 1.5

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FMC allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait-states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait-state (default after reset).
- The flash memory asserts the NWAIT signal during the wait-state

The FMC supports both NOR flash wait-state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0 to 3).

Figure 158. Wait configuration waveforms

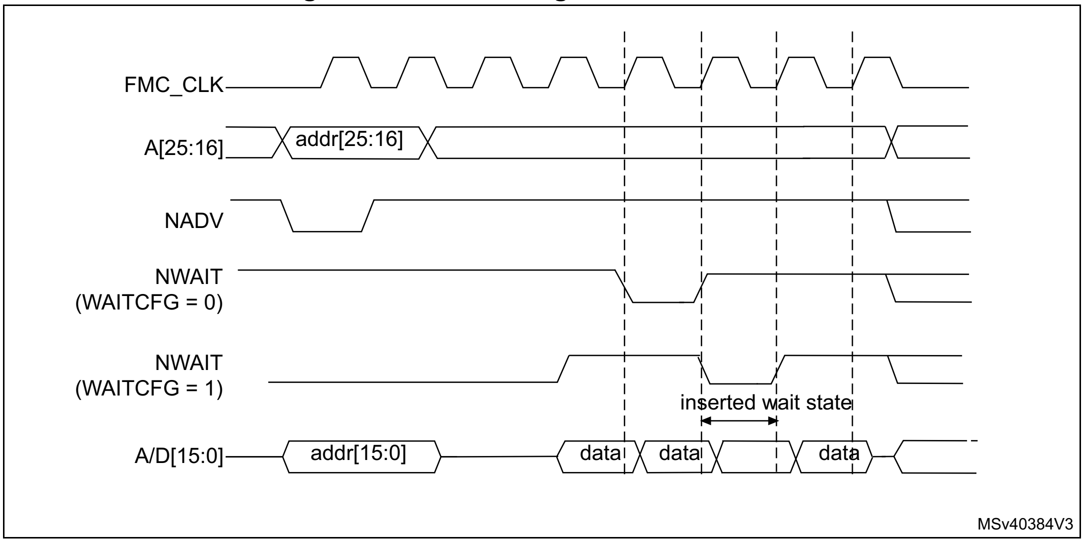


Figure 159. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)

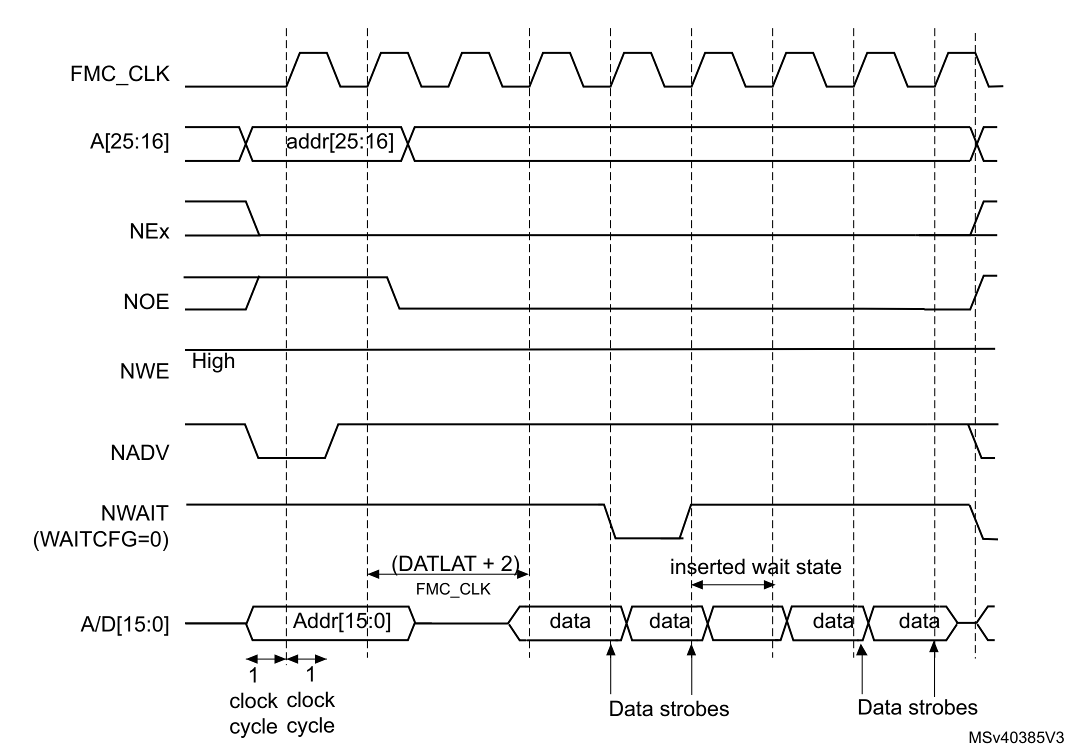


Table 163. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	As needed
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set if the memory supports this feature, to be kept cleared otherwise
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1

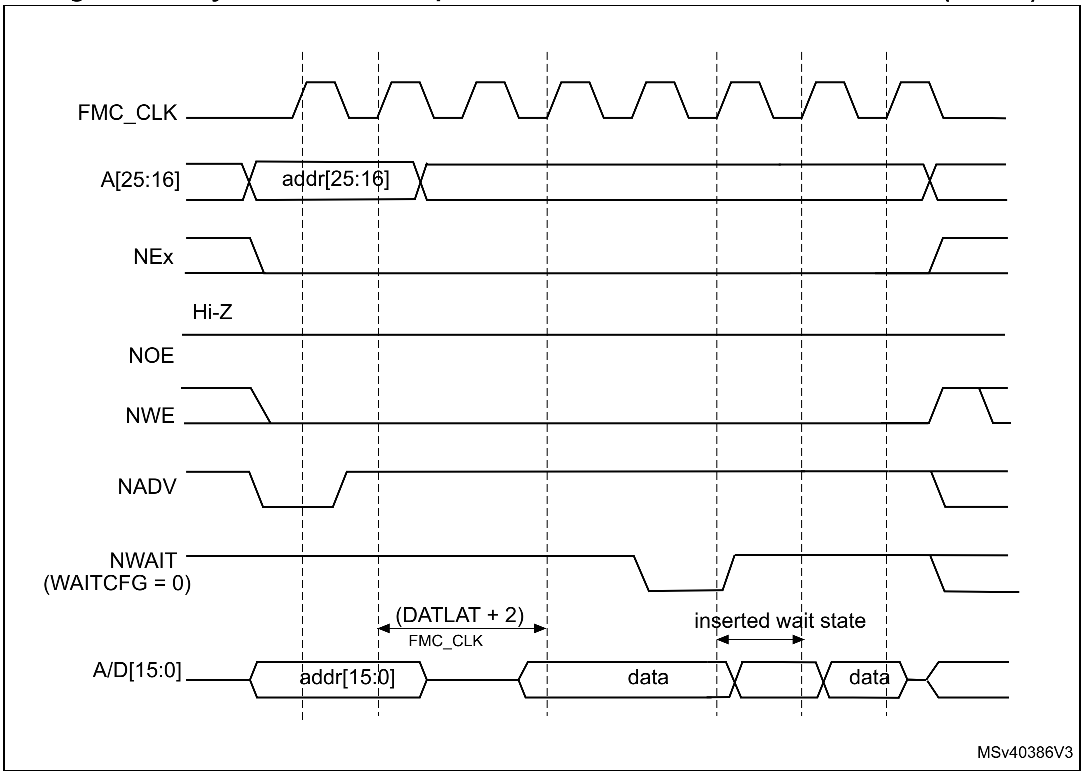
Table 163. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

Bit No.	Bit name	Value to set
6	FACCEN	Set according to memory support (NOR flash memory)
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 164. FMC_BTRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Don't care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	FMC_CLK divider ratio
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

Figure 160. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.

Table 165. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31	FMCEN	0x1
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21	Reserved	0x0
20	CCLKEN	As needed
19	CBURSTRW	As needed
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set if the memory supports this feature, to be kept cleared otherwise.
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1

Table 165. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit No.	Bit name	Value to set
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 166. FMC_BTRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	FMC_CLK divider ratio
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

25.8.6 NOR/PSRAM controller registers

The peripheral registers have to be accessed by words (32-bit)

SRAM/NOR flash chip-select control register for memory region x (FMC_BCRx)

Address offset: 0x00 + 0x08 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory region, used for SRAMs, PSRAM, FRAM and NOR flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	CCLKEN	CBURSTRW	CPSIZE[2:0]		
rw								rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **FMCEN**: FMC enable

This bit enables/disables the FMC.
0: FMC disabled
1: FMC enabled

Note: The FMCEN bit of the FMC_BCR2 to 4 registers is don’t care. It is only enabled through the FMC_BCR1 register.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:22 **NBLSET[1:0]**: Byte lane (NBL) setup

These bits configure the NBL setup timing from NBLx low to Chip select NEx low.
00: NBL setup time is 0 fmc_ker_ck clock cycle.
01: NBL setup time is 1 fmc_ker_ck clock cycle.
10: NBL setup time is 2 fmc_ker_ck clock cycles.
11: NBL setup time is 3 fmc_ker_ck clock cycles.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **CCLKEN:** Continuous clock enable

This bit enables the FMC_CLK clock output to external memory devices.

0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).

1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The FMC_CLK continuous clock is only enabled through the CCLKEN bit of the FMC_BCR1 register.

When CCLKEN bit is set, the CCLKEN bit of the FMC_BCR2 to 4 registers are don't care.

When FMC_CLK continuous clock is enabled, device 1 must be configured in synchronous mode (that is BURSTEN and/or CBURSTRW must be set) and the FMC_CLK clock ratio must be specified by CLKDIV in the FMC_BTR1 register. All the devices configured in synchronous mode are clocked by the same clock according to CLKDIV in FMC_BTR1. The CLKDIV in the FMC_BTR2 to 4 and FMC_BWTR1 to 4 registers have no effect.

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.

0: Write operations are always performed in asynchronous mode

1: Write operations are performed in synchronous mode.

Bits 18:16 **CPSIZE[2:0]:** CRAM page size

These bits are used for CellularRAM™ 1.5 which does not allow burst accesses to cross the address boundaries between pages. When these bits are configured, the FMC automatically splits the burst access when the memory page size is reached (refer to memory datasheet for page size).

000: No burst split when crossing page boundary. (default after reset).

001: 128 bytes

010: 256 bytes

011: 512 bytes

100: 1024 bytes

Others: Reserved.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables wait signal usage by the FMC even during asynchronous transfers.

0: NWAIT signal not taken in to account during asynchronous transfers (default after reset)

1: NWAIT signal taken in to account during asynchronous transfers

Bit 14 **EXTMOD:** Extended mode enable

This bit enables write timing programming for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: Values inside FMC_BWTR register not taken into account (default after reset)

1: Values inside FMC_BWTR register taken into account

Note: When the extended mode is disabled, the FMC can operate in mode 1 or mode 2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP[1:0] =0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP[1:0] = 0x10).*

Bit 13 **WAITEN:** Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in synchronous mode.

0: NWAIT signal disabled (its level not taken into account, no wait-state inserted after the programmed flash latency period)

1: NWAIT signal enabled (its level is taken into account after the programmed latency period to insert wait-states if asserted) (default after reset)

Bit 12 **WREN:** Write enable bit

This bit indicates whether write operations to the memory region by the FMC are enabled/disabled:

0: Write operations to the memory region by the FMC disabled. An AXI slave error is reported,

1: Write operations to the memory region by the FMC enabled (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait-state must be inserted when accessing the memory in synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait-state or during the wait-state:

0: NWAIT signal active one data cycle before wait-state (default after reset),

1: NWAIT signal active during wait-state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL:** Wait signal polarity bit

This bit defines the polarity of the wait signal from memory used for either in synchronous or asynchronous mode:

0: NWAIT active low (default after reset),

1: NWAIT active high.

Bit 8 **BURSTEN:** Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in burst mode:

0: Burst mode disabled (default after reset). Read accesses are performed in asynchronous mode.

1: Burst mode enable. Read accesses are performed in synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN:** Flash memory hardware reset

This bit enables the reset by hardware of the Flash memory, through a dedicated output pad.

0: the corresponding NOR flash memory reset output level is low

1: the corresponding NOR flash memory reset output level is high (default after reset)

Bits 5:4 **MWID[1:0]:** Memory data bus width

This bit defines the external memory device width, valid for all type of memories.

00: 8 bits

01: 16 bits (default after reset)

10: Reserved.

11: Reserved.

Bits 3:2 **MTYP[1:0]:** Memory type

This bit defines the type of external memory attached to the corresponding memory region:
00: SRAM/ FRAM (default after reset for memory region 2...4)
01: PSRAM (CRAM)/FRAM
10: NOR flash/OneNAND flash (default after reset for memory region 1)
11: Reserved.

Bit 1 **MUXEN:** Address/data multiplexing enable bit

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
0: Address/Data non-multiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory region enable bit

This bit enables the memory region. After reset memory region 1 is enabled, all others are disabled. Accessing a disabled memory region causes an ERROR on AXI bus.
0: Corresponding memory region is disabled
1: Corresponding memory region is enabled

SRAM/NOR flash chip-select timing registers for memory region x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory region, used for SRAMs, PSRAM and NOR flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, two registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data Hold phase duration

These bits are written by software to define the duration of the *data hold* phase in fmc_ker_ck cycles (refer to *Figure 156* to *Figure 157*), used in asynchronous accesses:

For read/write accesses:

00: DATAHLD phase duration = fmc_ker_ck clock cycle * 0 (default, read)/1 (default, write)

01: DATAHLD phase duration = fmc_ker_ck clock cycle * 1 (read)/2 (write)

10: DATAHLD phase duration = fmc_ker_ck clock cycle * 2 (read)/3 (write)

11: DATAHLD phase duration = fmc_ker_ck clock cycle * 3 (read)/4 (write)

Bits 29:28 **ACCMOD[1:0]:** Access mode

These bits specify the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: Access mode A

01: Access mode B

10: Access mode C

11: Access mode D

Bits 27:24 **DATLAT[3:0]:** Data latency for synchronous memory (see note below bit descriptions)

For synchronous access with read/write burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data.

This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.

For asynchronous access, this value is don't care.

0000: Data latency of 2 FMC_CLK clock cycles for first burst access

1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

These bits define FMC_CLK clock output signal period, expressed in fmc_ker_ck cycles:

0000: Reserved.

0001: FMC_CLK period = 2 × fmc_ker_ck periods

0010: FMC_CLK period = 3 × fmc_ker_ck periods

1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)

This value is don’t care when accessing NOR flash memories, SRAM and PSRAM in asynchronous mode.

Note: Refer to Section 25.8.5: Synchronous transactions for FMC_CLK *divider ratio formula).*

Bits 19:16 **BUSTURN[3:0]:** Bus turnaround phase duration

These bits are written by software to add a delay between the end of current read or write transaction and the next transaction on the same memory region.

This delay allows matching the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time required by the memory to free the data bus after a read access (t_{EHQZ} from chip enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to a minimum value that respects the following condition:

(BUSTRUN + 1)fmc_ker_ck period ≥ $t_{EHELmin}$ max ($t_{EHELmin}$, $t_{EHQZmax}$)

For FRAMs, the bus turnaround delay must be configured to match the minimum precharge time (t_{PC}). This delay is inserted between all consecutive transactions on the same memory region (read/read, write/write, read/write and write/read) to match the t_{PC} memory timing, and the Chip select toggles between consecutive accesses.

(BUSTRUN + 1) fmc_ker_ck period >= t_{PCmin}

0000: BUSTURN phase duration = 1 * fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 16 * fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 156* to *Figure 157*) used in asynchronous accesses:

0000 0000: Reserved.

0000 0001: DATAST phase duration = 1 * fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 * fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 * fmc_ker_ck clock cycles (default value after reset)

For each memory type and access mode data-phase duration, refer to the respective figure (*Figure 156* to *Figure 157*).

Example of mode 1, write access, DATAST=1:

Data-phase duration= DATAST+1 = 2 *fmc_ker_ck clock cycles.

Note: This value is don't care in synchronous accesses.

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 152* to *Figure 155*) used in mode D or multiplexed accesses:

0000: Reserved.

0001: ADDHLD phase duration =1 * fmc_ker_ck clock cycle

0010: ADDHLD phase duration = 2 * fmc_ker_ck clock cycle

...

1111: ADDHLD phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

For each access mode address-hold phase duration, refer to the respective figure (*Figure 152* to *Figure 155*).

Note: This value is not used in synchronous accesses, and the address hold phase lasts always 1 memory clock period.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 156* to *Figure 157*), used in SRAMs, ROMs and asynchronous NOR flash memories:

0000: ADDSET phase duration = 0 * fmc_ker_ck clock cycle

...

1111: ADDSET phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

For each access mode address setup phase duration, refer to the respective figure (refer to *Figure 156* to *Figure 157*).

Note: This value is don't care in synchronous accesses.

In muxed mode or mode D, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed.

On PSRAMs (CRAMs), DATLAT must be configured to 0 so that the FMC quickly exits its latency phase, starts sampling NWAIT from memory, and starts read or write operations when the memory is ready.

This method can be used also with the latest generation of synchronous flash memories, which, unlike older flash memories, issue the NWAIT signal (check the corresponding flash memory datasheet).

SRAM/NOR-flash write timing registers for memory region x (FMC_BWTRx)

Address offset: 0x104 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x000F FFFF

This register contains the control information of each memory region. It is used for SRAMs, FRAMs, PSRAMs and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN[3:0]			
rw	rw	rw	rw									rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data Hold phase duration

These bits are written by software to define the duration of the *data hold* phase in fmc_ker_ck cycles (refer to *Figure 157*), used in asynchronous write accesses:
00: DATAHLD phase duration = 1 * fmc_ker_ck clock cycle (default)
01: DATAHLD phase duration = 2 * fmc_ker_ck clock cycle
10: DATAHLD phase duration = 3 * fmc_ker_ck clock cycle
11: DATAHLD phase duration = 4 * fmc_ker_ck clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode.

These bits specify the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.
00: Access mode A
01: Access mode B
10: Access mode C
11: Access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay between the end of current write transaction and the next transaction on the same memory region.
For FRAMs, the bus turnaround delay must be configured to match the minimum precharge time (t_{PC}). The bus turnaround delay is inserted between all consecutive transactions on the same memory region (read/read, write/write, read/write and write/read), and the Chip select toggles between consecutive accesses.
(BUSTRUN + 1) fmc_ker_ck period >= t_{PCmin}
0000: BUSTURN phase duration = 1 * fmc_ker_ck clock cycle added
...
1111: BUSTURN phase duration = 16 * fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration.

These bits are written by software to define the data phase duration (refer to *Figure 156* to *Figure 157*) used in asynchronous SRAM, PSRAM and NOR flash memory accesses:

0000 0000: Reserved.

0000 0001: DATAST phase duration = 1 * fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 * fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 * fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration.

These bits are written by software to define the duration of the address hold phase (refer to *Figure 154* to *Figure 155*) used in asynchronous multiplexed accesses:

0000: Reserved.

0001: ADDHLD phase duration = 1 * fmc_ker_ck clock cycle

0010: ADDHLD phase duration = 2 * fmc_ker_ck clock cycle

...

1111: ADDHLD phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous NOR flash accesses, this value is not used, and the address hold phase always lasts 1 flash clock period.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration.

These bits are written by software to define the duration of the address setup phase in fmc_ker_ck cycles (refer to *Figure 156* to *Figure 157*) used in asynchronous accesses:

0000: ADDSET phase duration = 0 * fmc_ker_ck clock cycle

...

1111: ADDSET phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, and the address setup phase always lasts 1 flash clock period. In muxed mode, the minimum ADDSET value is 1.

PSRAM chip select counter register (FMC_PCSCNTR)

Address offset: 0x20

Reset value: 0x0000 0000

This register contains the PSRAM chip select counter value for synchronous mode. The chip select counter is common to all memory regions and can be enabled separately on each memory region. During PSRAM read or write accesses, this value is loaded into a timer which is decremented using the fmc_ker_ck while the NE signal is held low. When the timer reaches 0, the PSRAM controller splits the current access, toggles NE to allow PSRAM device refresh and restarts a new access. The programmed counter value guarantees a maximum NE pulse width (t_{CEM}) as specified for PSRAM devices. The counter is reloaded and starts decrementing each time a new access is started by a transition of NE from high to low.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTB4EN	CNTB3EN	CNTB2EN	CNTB1EN
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSCOUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CNTB4EN:** Counter memory region 4 enable
This bit enables the CS counter for PSRAM/NOR memory region 4.
0: Counter disabled for memory region 4
1: Counter enabled for memory region 4

Bit 18 **CNTB3EN:** Counter memory region 3 enable
This bit enables the CS counter for PSRAM/NOR memory region 3.
0: Counter disabled for memory region 3.
1: Counter enabled for memory region 3

Bit 17 **CNTB2EN:** Counter memory region 2 enable
This bit enables the CS counter for PSRAM/NOR memory region 2.
0: Counter disabled for memory region 2
1: Counter enabled for memory region 2

Bit 16 **CNTB1EN:** Counter memory region 1 enable
This bit enables the CS counter for PSRAM/NOR memory region 1.
0: Counter disabled for memory region 1
1: Counter enabled for memory region 1

Bits 15:0 **CSCOUNT[15:0]:** Chip Select (CS) counter.
These bits are written by software to define the maximum chip select low pulse duration (in fmc_ker_ck cycles) for synchronous accesses.
The counter is disabled if the programmed value is 0.
The number of fmc_ker_ck clock cycles versus the programmed value is obtained by the below equation:
number of fmc_ker_ck cycles = (CLKDIV+1)*ROUNDUP(CSCOUNT[15:0]/(CLKDIV+1);0)

FMC size identification register (FMC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size Identification
These bits return the size of the memory region allocated to the FMC registers.

FMC identification register (FMC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0014 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** FMC Identifier
These bits return the FMC identifier

FMC version register (FMC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** Major revision
These bits return the FMC major revision

Bits 3:0 **MINREV[3:0]:** Minor revision
These bits return the FMC minor revision

FMC hardware configuration register 1 (FMC_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x2232 B011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA_LN2DPTH[3:0]				WR_LN2DPTH[3:0]				WD_LN2DPTH[3:0]				WA_LN2DPTH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_SIZE[3:0]				Res.	Res.	Res.	SDRAM_SEL	Res.	Res.	Res.	NAND_ECC	Res.	Res.	Res.	NAND_SEL
r	r	r	r				r				r				r

Bits 31:28 **RA_LN2DPTH[3:0]:** AXI read address FIFO depth

These bits return the AXI read address FIFO depth. depth is $2^{RA\_LN2DPTH}$.

Bits 27:24 **WR_LN2DPTH[3:0]:** AXI write response FIFO depth

These bits return the AXI write response FIFO depth. depth is $2^{WR\_LN2DPTH}$.

Bits 23:20 **WD_LN2DPTH[3:0]:** AXI write data FIFO depth

These bits return the AXI write data FIFO depth. depth is $2^{WD\_LN2DPTH}$.

Bits 19:16 **WA_LN2DPTH[3:0]:** AXI write address FIFO depth

These bits return the AXI write address FIFO depth. depth is $2^{WA\_LN2DPTH}$.

Bits 15:12 **ID_SIZE[3:0]:** AXI ID width

These bits return the AXI ID width.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDRAM_SEL:** SDRAM controller selection

0x0: SDRAM controller not supported
0x1: SDRAM controller supported

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **NAND_ECC:** NAND ECC

0x0: Hamming code (supported only in Legacy mode without sequencer)
0x1: Hamming and BCH code

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **NAND_SEL:** NAND Controller Selection

0x0: NAND Controller not supported
0x1: NAND Controller supported

FMC hardware configuration register 2 (FMC_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x00DC 8762

The base addresses are relevant only when the FMC includes the corresponding memory controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDRAM2_BASE[3:0]				SDRAM1_BASE[3:0]			
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAND_BASE[3:0]				SDRAM_RBASE[3:0]				NOR_BASE[3:0]				RD_LN2DPTH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **SDRAM2_BASE[3:0]:** SDRAM bank 2 base address
It returns bits [31:28] of SDRAM bank 2 base address space when the SDRAM controller is supported. It is ignored otherwise.

Bits 19:16 **SDRAM1_BASE[3:0]:** SDRAM bank 1 base address
It returns bits [31:28] of SDRAM bank 1 base address space when the SDRAM controller is supported. It is ignored otherwise.

Bits 15:12 **NAND_BASE[3:0]:** NAND base address
It returns bits [31:28] of NAND base address space

Bits 11:8 **SDRAM_RBASE[3:0]:** SDRAM remap base address
it returns bits [31:28] of SDRAM remap base address space when the SDRAM controller is supported. It is ignored otherwise.

Bits 7:4 **NOR_BASE[3:0]:** NOR base address
It returns bits [31:28] of NOR base address space

Bits 3:0 **RD_LN2DPTH[3:0]:** AXI read data FIFO depth
These bits return the AXI read data FIFO depth. depth is $2^{RD\_LN2DPTH}$

25.9 NAND flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND flash memories

The NAND memory region is configured through dedicated registers (*Section 25.9.9*). The programmable memory parameters include access timings (shown in *Table 167*) and ECC configuration.

The NAND controller supports the following features:

- Programmable page size of 256, 512, 1024, 2048, 4096 or 8192 bytes
- Programmable memory timings
- Programmable Error Correction Capability using BCH code or Hamming code
- Command sequencer: a hardware accelerator for read/write accesses within a page or within a block of configurable size.
  This feature is useful when ECC computation is performed.
- Programmable address offset for ECC bytes programming using the command sequencer
- Interrupt generation on ECC error detection and page or sector transfer complete
- Support of multiple dice per package. Only one die can be selected at a time.

Table 167. Programmable NAND flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory wait	Minimum duration (in fmc_ker_ck clock cycles) of the command assertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory hold	Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command deassertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory databus high-Z	Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started	Write	FMC clock cycle (fmc_ker_ck)	1	256

25.9.1 External memory interface signals

The following tables list the typical signals that are used to interface NAND flash memory.

Note: The "N" prefix identifies the signals which are active low.

8-bit NAND flash memory

Table 168. 8-bit NAND flash memory

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCEx	O	Chip Select (x=1, 2)
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT	I	NAND flash Ready/Busy (RNB) input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND flash memory

Table 169. 16-bit NAND flash memory

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCEx	O	Chip Select (x=1, 2)
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT	I	NAND flash Ready/Busy (RNB) input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

25.9.2 NAND flash supported memories and transactions

Table 170 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.

Table 170. Supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	
	Asynchronous	W	8	8	Y	
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
	Asynchronous	R	64	8	Y	Split into 8 FMC accesses
	Asynchronous	W	64	8	Y	Split into 8 FMC accesses
NAND 16-bit	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses

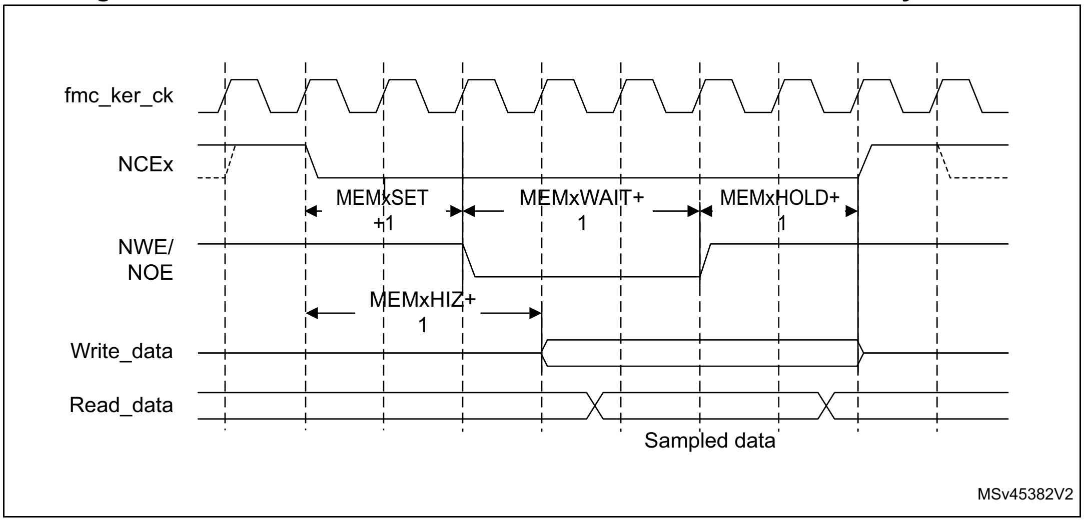
25.9.3 Timing diagrams for NAND flash memory

The NAND flash memory region is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_HECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

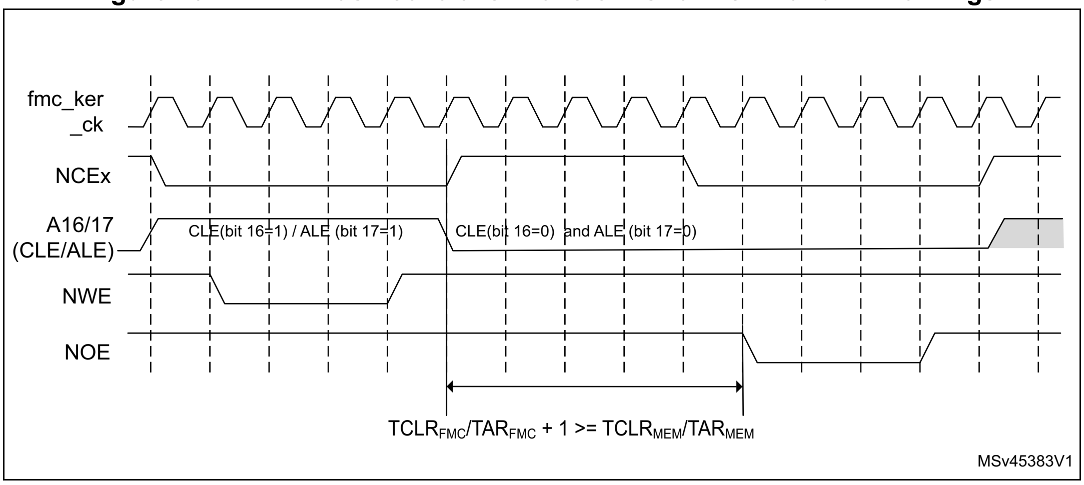
Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the different phases of any NAND flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. *Figure 161* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 161. NAND flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. If PWAITEN = 1, the programmed values of (MEMxSET + MEMxHOLD) must be > 1.

Figure 162. NAND flash controller waveforms for TCLR and TAR timings



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.

25.9.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the NAND memory region by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 25.9.5: NAND flash prewait function*) registers according to the characteristics of the NAND flash memory (PWID bits to configure NAND flash data bus width, PWAITEN set or cleared as needed, see *Section 25.7.2: NAND flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with a data byte value equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The NAND flash CLE input is active during the write strobe (low pulse on NWE), so that the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address for a read operation by writing a five-byte address to the common memory or the attribute space followed by the second read command. The NAND flash ALE input is active during the write strobe (low pulse on NWE), so that the written bytes are interpreted as the start address for read operations. Using the attribute memory space allows to use a different FMC timing configuration, which can be used to implement the prewait function required for some NAND flash memories (see details in *Section 25.9.5: NAND flash prewait function*).
4. The controller waits till the NAND flash memory is ready (RNB signal high), before starting a new access to the same memory region or to another memory region.
5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.
6. No CPU command or address write operation is required to read the next NAND flash page. This can be done in three different ways:
   - by simply performing the operation described in step 5,
   - a new random address can be accessed by restarting the operation at step 3,
   - a new command can be sent to the NAND flash device by restarting at step 2.

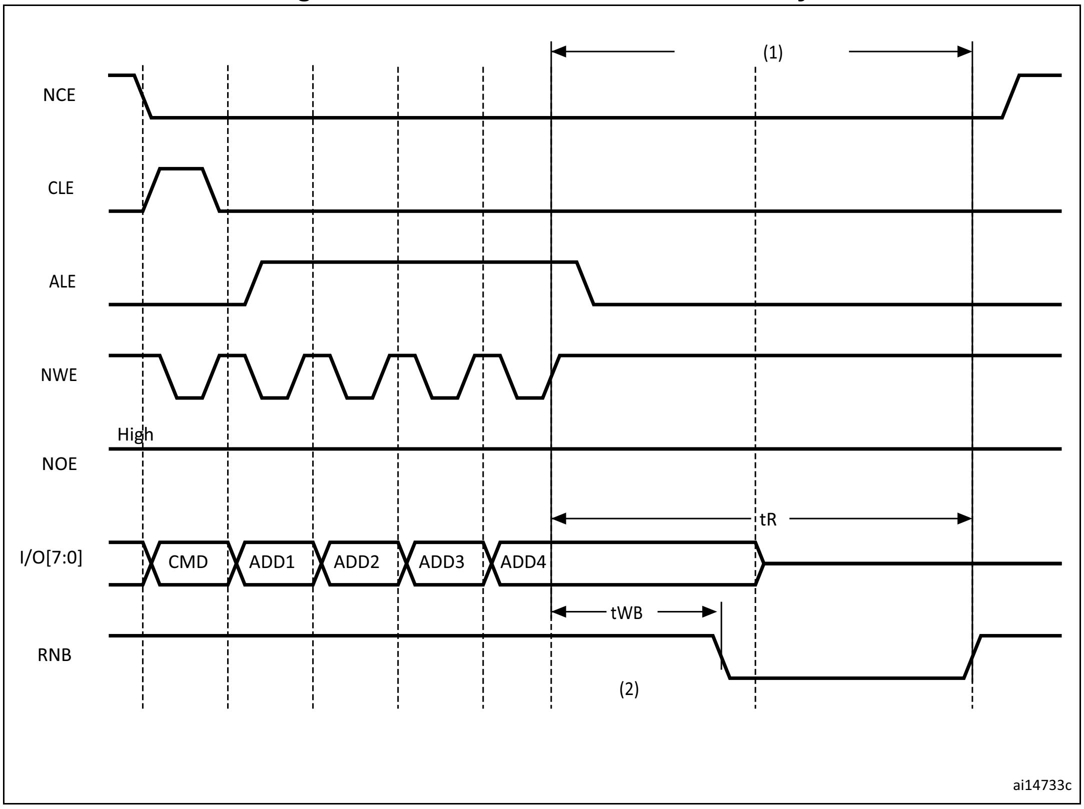
Note: Non-‘CE don’t care’ NAND flash memories are not supported.

ONFI mode 4 and 5 EDO data output cycle timings are not supported.

25.9.5 NAND flash prewait function

The NAND controller must wait for RNB to be asserted low before sending its next command (see *Figure 163*). This timing constraint is set according to t_{WB} using MEMHOLD[7:0] bitfield in FMC_PMEM or ATTHOLD[7:0] in FMC_PATT.

Figure 163. Access to NAND flash memory



1. NCE can be deasserted for NAND flash memories since CE is don't care except for very old NAND flash devices that are not considered in this example.
2. RNB sampling can be delayed after t_{WB} by using ATTHOLD[7:0] bit of FMC_PATT. As an example, if fmc_ker_clk = 14 ns and ATTHOLD[7:0] = 7, the delay is 8*14 = 112 ns. The usual t_{WB} maximum value is 100 ns and the programmed delay must be higher than t_{WB} + 2*fmc_ker_clk.

Any CPU read or write access to the NAND flash memory has a delay of (MEMHOLD[7:0] + 1) * fmc_ker_ck cycles after the NWE signal is asserted.

The ATTHOLD[7:0] bitfield in FMC_PATT register must be determined to satisfy t_{WB} timing constraints and selectively apply larger timing constraints to the last command or address byte in CPU mode. When setting ATTHOLD[7:0], the 2 * fmc_ker_ck delay required for the sampling of RNB by NAND sequencer must also taken into account.

25.9.6 NAND ECC controller

NAND flash memories include a spare memory area (or OOB) at the end of each page. This area is typically used to store some metadata (such as Bad Block Marker information to invalidate the whole page) or for error management functions.

The FMC supports on-the-fly error correction code (ECC) computation during data read/program from/to NAND flash memory. This mechanism reduces the host CPU workload when processing the ECC by software.

The user application can select one out of two algorithms:

- Hamming code algorithm for 1-bit error code correction and 2-bit error detection per 256, 512, 1024, 2048, 4096 or 8192 bytes.
- BCH (Bose, Chaudhuri and Hocquenghem) code for 4-bit error code correction and 8-bit error detection or 8-bit error code correction and 16-bit error detection per 512 bytes.

The principle of both ECC controllers is similar: the ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory region is active. The ECC controllers operate as follows:

- When accessing a NAND flash memory region, the data present on the D[7:0] or D[15:0] bus are latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU or DMA, the ECC result registers are updated.

Hamming code

The Hamming code implemented can perform 1-bit error correction and 2-bit error detection per 256, 512, 1024, 2048, 4 096 or 8192 bytes read or written from/to the NAND flash memory. It consists in calculating the row and column parity. This algorithm is supported by 8-bit and 16-bit NAND flash memories.

Page write and read sequences with Hamming code are described below.

To perform a page program with Hamming ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC controller computes the ECC value.
3. Wait until the ECC code is ready (NWRF is set in the FMC_SR register).
4. Read the ECC value available in the FMC_HECCR register and store it in NAND flash spare area.
5. Clear the ECCEN bit and launch the page programming.

To perform a page read with Hamming ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register before reading the NAND page. While the NAND page is read, the ECC block computes the ECC value.
2. Wait until the ECC code is ready (NWRF set in the FMC_SR register).
3. Read the ECC value available in the spare area as well also the ECC value, processed by Hamming controller, available in FMC_HECCR register.
4. If the two ECC values are identical, no correction is required. Otherwise an ECC error occurred, the software correction routine returns information on whether the error can be corrected or not, and corrects it when possible.

Depending on the spare area size, the page can be split into several sectors of identical size. This allows increasing the error correction capability for each page since 1 error can be corrected and 2 errors can be detected for each sector.

Example

For a NAND flash memory with 2 Kbytes + 64 bytes pages, each page can be split into 8 sectors of 256 bytes each. 3 bytes (22 bits) of ECC are required for each sector, which makes 24 bytes (8 * 3 bytes) to be stored in the spare area for the whole page. For the whole 2 Kbyte page, up to 8 errors can be corrected, respectively up to 16 errors detected, provided there is less than 1 error, respectively 2 errors, per sector.

BCH (Bose, Chaudhuri and Hocquenghem) code

To increase the error correction capability, the FMC embeds a BCH (Bose, Chaudhuri and Hocquenghem) encoder and decoder.

It supports 4-bit error code correction and 8-bit error detection or 8-bit error code correction, and 16-bit error detection per sector. BCH encoder/decoder module handles sectors of fixed size, equal to 512 bytes. Each page must consequently be split into 512-byte sectors. The number of parity bytes generated for each sector is given in *Table 171*. For 16-bit NAND flash memories, a padding is added to align parity bytes to an even number (8 bytes and 14 bytes for 4-bit and 8-bit BCH code, respectively).

Table 171. Number of ECC parity bytes per sector

Function	BCH code	
Error correction	4-bit	8-bit
Number of parity bytes	7 bytes	13 bytes

The BCH module detects errors and returns the bit position for each sector. To correct the errors, the software needs to flip the error bits at the provided bit position stored in the FMC_BCHPBRx register.

The BCH ECC controller supports 8-bit and 16-bit NAND flash memories.

During a page program operation, the BCH encoder generates the ECC bytes (7 or 13 bytes) for each sector of 512 bytes that must be written in the NAND flash spare area with a programmable offset. It must take into account some metadata at the beginning of the spare area. This operation must be executed for each sector. The BCH encoder does not process the spare area: metadata must have their own error protection mechanism.

The total number of ECC bytes (N) is given by the below equation:

$$N = \text{Number of ECC bytes /sector} \times \text{Number of sector/page}$$

Example

For an 8-bit NAND flash with 4096-byte pages and an error correction capability of 4-bit per 512 bytes, the number of sector per page is 8 (4096 / 512). The total number of ECC bytes is consequently:

$$N = 7 \times 8 = 56 \text{ bytes}$$

The user application has to make sure that the spare area is large enough to store the metadata and parity bytes.

Page program and read sequences with BCH code are described below.

To perform a page program:

1. Enable write access by setting the WEN bit in the FMC_PCR register.
2. Enable the ECC by setting the ECCEN bit in FMC_PCR register.
3. Write 512 data bytes to the sector in the NAND flash memory page. While the sector is written, the BCH encoder computes the ECC value.
4. Wait until the BCH encoder parity bits are available by monitoring EPBRF bit of FMC_BCHISR register.
5. Read the ECC value available in the FMC_BCHPBRx registers and store it in the NAND flash spare area.
6. Disable ECC by resetting the ECCEN bit in FMC_PCR register.
7. Execute again step 2 to 6 for all sectors to be written in NAND flash memory.
8. Launch the page programming.

To perform a page read:

1. Enable read access by resetting the WEN bit in the FMC_PCR register.
2. Enable the ECC by stetting the ECCEN bit in FMC_PCR register.
3. Read the NAND page sector (512 bytes).
4. Read the corresponding ECC bytes (7 or 13 bytes) from the NAND flash spare area.
5. During the read of data and parity bits, the syndrome is calculated. Then the error location search is launched.
6. Wait until the decoding error is ready (DERF bit set in the FMC_BCHISR register), then read the decoding results (errors detected if any, correctable or not, number of errors, bit position) in FMC_BCHDSR0 and FMC_BCHDSRx registers.
7. Thanks to the decoding results, the software can correct the errors in the sector (when possible).
8. Disable ECC by resetting the ECCEN bit in FMC_PCR register.
9. Execute again step 2 to 7 for all relevant sectors of the page.

25.9.7 FMC command sequencer

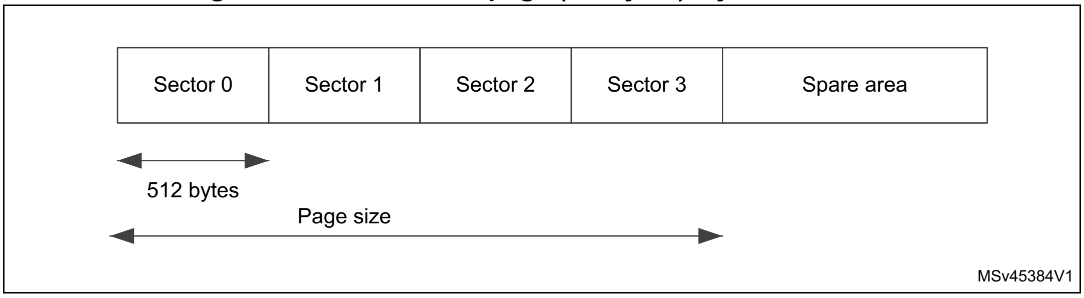
NAND flash page programming and reading can be performed by the CPU as described in the above sections.
Data sector programming/reading as well as ECC byte programming/reading can also be managed automatically by the FMC command sequencer for command and address transfers, and by DMA for data transfers.

The purpose of the command sequencer is to ease the programming/reading of NAND flash pages with ECC computation as well as free the CPU of sequencing tasks. The command sequencer enables the software to work on full NAND flash pages instead of sectors, which is more efficient and essential to achieve high throughput. The sequencer handles all the sectors in the page. It does not support sub-page read or program operations.

One DMA channel is required for write operations and two DMA channels for read operations.

A specific NAND flash page layout must be respected when using the BCH algorithm. See *Figure 164* for an example of 2Kbyte page BCH layout.

Figure 164. NAND flash page (2 Kbytes) layout with BCH



Page program sequence

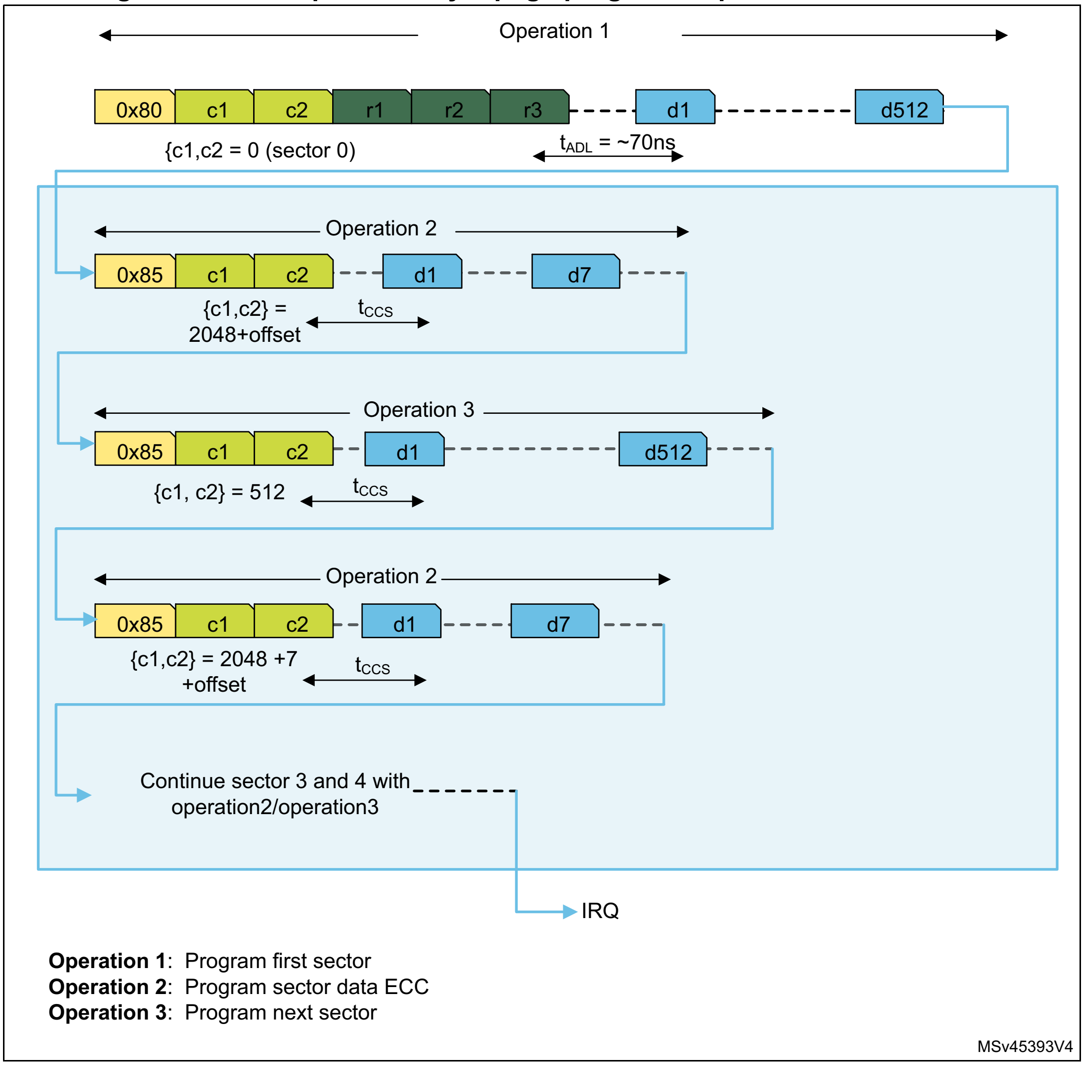
1. A data buffer located in SRAM has to be initialized with the data to be written to the NAND flash page.
2. The CPU configures the FMC command sequencer registers: page program command (0x80), address bytes, change write column command (0x85), ECC address offset, page size, sector size and sector number. It also selects the chip enable.

   It also selects the ECC algorithm (Hamming or BCH) and configures one DMA channel to transfer data from the buffer to the NAND flash page.
3. The CPU then launches the sequencer by setting the CSQSTART bit in the FMC_CSQCR register.
4. The command sequencer automatically generates the control and address cycles to the NAND flash memory, following the programmed timings configured either in FMC_PMEM or FMC_PATT register.
5. A DMA request is issued to start data transfer from the data buffer to NAND flash memory.
6. When the full sector is programmed, go to step 7 if ECC is enabled. Otherwise, jump to step 9.
7. The FMC command sequencer automatically issues a change write column command (0x85) followed by two address cycles. The command to be issued and the number of address cycles are programmed in FMC_CSQCFGR2 register. The address is automatically generated from the programmed ECC address offset.
8. Once the parity bytes are available in ECC encoder (Hamming or BCH) module, they are automatically transferred to the NAND flash spare area. The FMC command sequencer can generate a sector transfer complete interrupt if it has been previously enabled.
9. The sequencer then repeats step 4 (except that it performs only a change write column command instead of a page program command) to step 8 to program the following sectors and their corresponding ECC parity bytes. The ECC address offset is automatically incremented for each sector transfer. When ECC is disabled, the sequencer repeats step 5.
10. The FMC command sequencer can generate a transfer complete interrupt or a transfer complete error interrupt by sector or when all sectors are written.
11. The CPU can program any further metadata in the NAND flash area, taking care not to overwrite parity bytes.
12. The program command (0x10) can then be written to the command section to program the NAND flash memory.

13. An interrupt, if enabled, is generated on the rising edge of the Ready/Busy (RNB) signal to indicate that the page program operation is complete.

Refer to *Figure 165* for an overview of the page program sequence with BCH.

Figure 165. Example of 2 Kbyte page program sequence with 4-bit BCH



Page read sequence

1. A data buffer is allocated in SRAM to store the NAND flash page data. A decoding status buffer is also allocated to store result of ECC decoding.
2. The CPU programs the command sequencer registers: the page read first cycle command (0x00), address bytes, the page read second cycle command (0x30), change read column command (0x05), spare area offset, page size, sector size and sector number. It also selects the chip enable.

   It configures the ECC algorithm if required (Hamming or BCH), one DMA channel to handle data transfers from the NAND flash page to the data buffer, and a second DMA channel to transfer ECC decoding results from ECC status registers to the RAM decoding status buffer.

3. The CPU then launches the FMC sequencer by setting the CSQSTART bit in the FMC_CSQCR register.
4. The FMC command sequencer issues the command and address cycles to the NAND flash memory.
5. The command sequencer waits for a Ready/Busy (RNB) rising edge to start data transfer.
6. It sends a DMA request to launch the data transfer from the NAND flash page to the data buffer.
7. When all sector data are read, go to step 8 if ECC is enabled, otherwise go to step 9. The FMC command sequencer can generate a sector transfer complete interrupt if it has been previously enabled.
8. When ECC status registers are empty (because the Hamming error code or the BCH error position bits of previous sector have already been read), the ECC command sequencer automatically issues a change read column command (0xE0) followed by two address cycles and by the 0xE0 command, to read ECC bytes from NAND flash memory and store them in ECC registers. These commands as well as the number of address cycles are programmed in the FMC_CSQCFGR2 register (see *Figure 166*).

   The address in spare area is extracted from the programmed ECC address offset.

   When the Hamming algorithm is used, a DMA request to the second DMA channel is send to transfer ECC codes (FMC_HPR and FMC_HECCR registers) to the decoding status buffer.

   When BCH algorithm is used, once all the parity bytes have been read, the BCH decoder first processes the syndrome to detect errors and then searches for error positions if any.

   As the BCH decoder is pipelined, it is possible to start reading a sector even if the decoding of the previous sector is not complete.

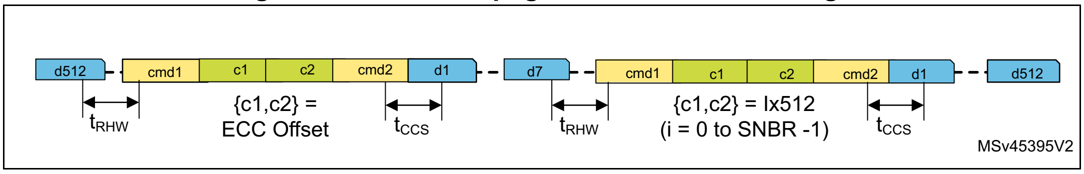
   In parallel of the following steps, when BCH sector decoding is complete, the new error status is stored in the ECC status registers (FMC_BCHDSR0) and a DMA request to the second DMA channel is sent to transfer data from the status registers to the decoding status buffer located in RAM.
9. The FMC command sequencer then repeats step 4 (except that a change read column command is issued instead of the page read first cycle command) to step 8 to read the other sectors and their corresponding ECC bytes. The addresses are issued and incremented following the page size, number of sectors, spare area address offset address and BCH code selection. When the read sequence is complete, the command sequencer waits for t_{RHWL} before issuing a new command sequence (refer to *Figure 165*). When ECC is disabled, the sequencer repeats step 6.
10. Once all sectors within a page and the status registers (FMC_BCHDSRx) are transferred by the command sequencer, the ECC controller can generate the transfer complete interrupt or transfer complete error (when enabled).
11. By reading the decoding status buffer, the CPU can start processing the errors corresponding to each sector. If no error has been detected, the page stored in RAM is validated. Otherwise the CPU corrects the error(s) before validating the page buffer in RAM. If there are too many errors, the CPU can detect the page cannot be corrected and invalidate the data buffer.

Following the selected BCH code, if the number of detected errors exceeds the number of correctable errors, this bit is not significant and the DUE bit is set. If the number of errors the detected by the BCH decoder exceeds the maximum number of detectable errors, DEN and DUE bits are not significant.

The DMA channels can also send interrupts each time a data transfer is complete. This allows the CPU to process error(s) in a sector as soon as it is available.

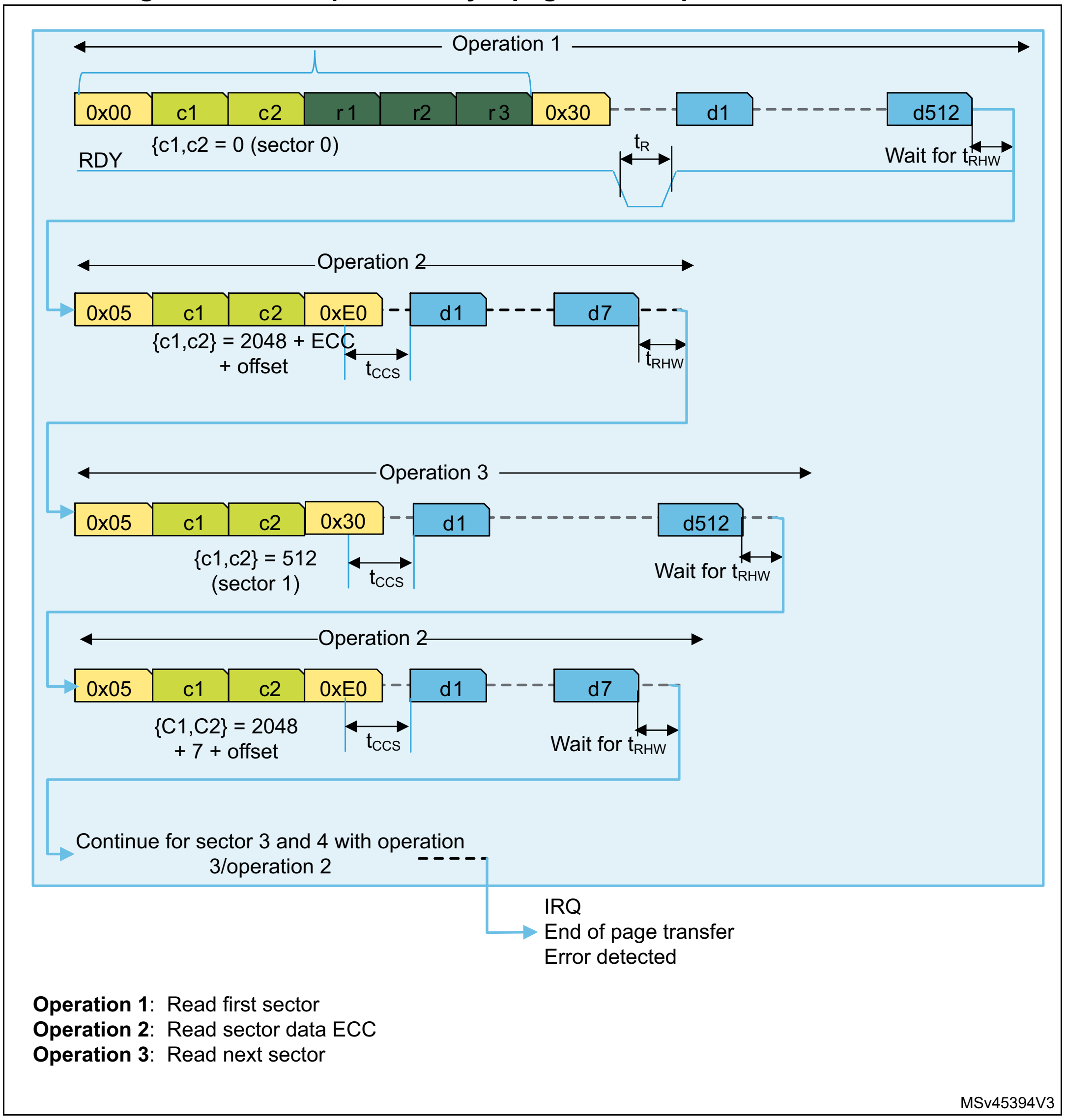
Note: When using the sequencer and DMA channels, the CPU must not read the FMC_BCHPBRx and FMC_BCHDSRx registers before data of all sectors are transferred, that is when TCF flag is set in the FMC_CSQISR register.

Figure 166. Chained-page read command timings



An example of 2 Kbyte page read sequence with BCH is described in *Figure 167*. This example corresponds to four 512-byte sectors and 7 ECC bytes per data 512 bytes.

Figure 167. Example of 2-Kbyte page read sequence with 4-bit BCH



The BCH controller resets the ECC computation whenever a page read (0x00), a page program (0x80), a random read (0x05) or a random program command ((0x85) is issued.

When only part of a page is written/read using the BCH, the command sequencer issues a random read/write command sequence only for the number of sectors to be read/written. This is done by programming the number of sector to be processed.

FMC command sequencer configuration

1. Configure the FMC_CSQCFGR1 register: program the command1 and the command 2 (in case of read access), the number of address cycles to be issued, configure the read or write sequence, and select the timings to be used.
2. Configure the address bytes in FMC_CSQAR1 and FMC_CSQAR2 register.
3. In the FMC_CSQCFGR2 register, configure the change write or change read column commands and the timings to be used for ECC. Program the address offset for the random columns in the FMC_CSQAR2 register. Enable the DMA if it is needed for ECC status register transfer.
4. Configure the number of sectors and the address cycle timings in the FMC_CSQCFGR3 register.
5. Configure the ECC sector size, and select read or write access in the FMC_PCR register.
6. Enable the SQSDTEN bit in the FMC_CSQFCGR2 register. The ECCEN bit in the FMC_PCR register kept at 0.
7. Transfer complete, and transfer error complete or sector transfer complete can be enabled through the FMC_CSQIER register.

25.9.8 NAND flash controller interrupt

An interrupt can be generated on the following events:

- RNB rising edge: an interrupt is generated when a rising edge is detected on RNB NAND flash signal
- RNB falling edge: an interrupt is generated when a falling edge is detected on RNB NAND flash signal
- RNB high-level: an interrupt is generated when a high-level is detected on RNB NAND flash signal.
- Transfer complete: an interrupt is generated when the FMC command sequencer has completed all data transfers for all sectors or when the command sequencer was aborted.
- Sector complete interrupt: an interrupt is generated when the FMC command sequencer has completed all data transfers for one sector.
- Sector error interrupt: an interrupt is generated when the FMC command sequencer has completed a sector transfer with at least one error detected.
- Sector uncorrectable error interrupt: an interrupt is generated when the FMC command sequencer has completed a sector transfer and detected an uncorrectable error.
- Command transfer complete: the command sequencer has completed the transfer of programmed commands and addresses.
- BCH encoder parity bits ready: an interrupt is generated when the BCH encoder has finished processing parity bits.
- BCH decoder syndrome ready: an interrupt is generated when the BCH decoder has finished processing the syndrome and knows if errors have been detected or not.
- BCH decoder error found: an interrupt is generated when the BCH decoder has detected an error.
- BCH decoder error ready: an interrupt is generated when the BCH decoder has finished processing errors, has identified error locations and can then correct them.
- BCH decoder uncorrectable error: an interrupt is generated when the BCH decoder has detected that the errors cannot be corrected (more than four or eight errors according to BCH error correction capability configuration).

The interrupt source can be enabled or disabled separately through the FMC_ IER, FMC_CSQIER and FMC_BCHIER registers.

Table 172. FMC NAND controller Interrupt request

Interrupt event	Event Flag	Enable control bit
Transfer complete	TCF in FMC_CSQISR	TCIE in FMC_CSQIER
Sector complete	SCF in FMC_CSQISR	SCIE in FMC_CSQIER
Sector error	SEF in FMC_CSQISR	SEIE in FMC_CSQIER
Sector uncorrectable error	SUEF in FMC_CSQISR	SUEIE in FMC_CSQIER
Command transfer complete	CMDTCF in FMC_CSQISR	CMDTCIE in FMC_CSQIER
BCH encoder parity bits ready	EPBRF in FMC_BCHISR	EPBRIE in FMC_BCHIER
BCH decoder syndrome ready	DSRF in FMC_BCHISR	DSRIE in FMC_BCHIER
BCH decoder error found	DEFF in FMC_BCHISR	DEFIE in FMC_BCHIER

Table 172. FMC NAND controller Interrupt request (continued)

Interrupt event	Event Flag	Enable control bit
BCH decoder error ready	DERF in FMC_BCHISR	DERIE in FMC_BCHIER
BCH decoder uncorrectable error	DUEF in FMC_BCHISR	DUEIE in FMC_BCHIER

25.9.9 NAND flash controller registers

NAND flash programmable control register (FMC_PCR)

Address offset: 0x80

Reset value: 0x0007 FE08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WEN	BCHECC	Res.	Res.	Res.	Res.	ECCSS[2:0]			TAR [3]
						rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				ECCALG	Res.	ECCEN	PWID[1:0]		Res.	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **WEN:** Write enable

This bit enables read or write access. It must be configured when using the FMC sequencer or when the ECC with BCH code is enabled. It can only be modified when ECCEN bit is reset.

0: Read access enabled

1: Write access enabled

Bit 24 **BCHECC:** BCH error correction capability

0: 4-bit BCH (4-bit error correction and 8-bit error detection per 512 bytes)

1: 8-bit BCH (8-bit error correction and 16-bit error detection per 512 bytes)

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCSS[2:0]:** ECC sector size (used to access spare area)

These bits define the sector size used for the ECC:

000: 256 bytes

001: 512 bytes

010: 1024 bytes

011: 2048 bytes (default)

100: 4096 bytes

101: Reserved.

11X: Reserved.

When the ECC is enabled with BCH code, the sector size must be programmed to 512 bytes.

Bits 16:13 **TAR[3:0]:** ALE to RE delay.

These bits define the time from ALE low to RE low in number of fmc_ker_ck clock cycles:
$t\_ar \leq (TAR + 1) \times t_{fmc\_ker\_ck}$
where $t_{fmc\_ker\_ck}$ is the FMC clock period and *t*_ar is the ALE to RE timing of NAND flash memories.
0000: 1 * fmc_ker_ck cycle
1111: 16 * fmc_ker_ck cycles (default)

Bits 12:9 **TCLR[3:0]:** CLE to RE delay.

These bits define the time from CLE low to RE low in number of fmc_ker_ck clock cycles:
$t\_clr \leq (TCLR + 1) \times t_{fmc\_ker\_ck}$
where $t_{fmc\_ker\_ck}$ is the FMC clock period and *t*_clr is the CLE to RE timing of NAND flash memories.
0000: 1 * fmc_ker_ck cycle
1111: 16 * fmc_ker_ck cycles (default)

Bit 8 **ECCALG:** ECC algorithm

0: Hamming code is selected (default).
1: BCH code is selected.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit

0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.
This bit must be kept reset when using the FMC sequencer

Bits 5:4 **PWID[1:0]:** Data bus width

These bits define the external memory device width.
00: 8 bits (default after reset).
01: 16 bits
1X: Reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PBKEN:** NAND flash memory region enable bit

This bit enables the memory region. Accessing a disabled memory region causes an AXI bus error.
0: Corresponding memory region is disabled (default after reset)
1: Corresponding memory region is enabled.

Note: The NAND flash memory region must be disabled only when all NAND flash controller commands are complete.

Bit 1 **PWAITEN:** Wait feature enable bit

This bit enables the Wait feature for the NAND flash memory region:
0: disabled (default)
1: enabled

Bit 0 Reserved, must be kept at reset value.

FMC status register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0053

This register contains information about the AXI interface isolation status and the NAND write requests status. The FMC has to be disabled before modifying some registers. As requests might be pending, it is necessary to wait till the AXI interface is stable and the core of the block is totally isolated from its AXI interface before reconfiguring the registers.

The PEF and NWEF bits indicate the status of the pipe. If Hamming algorithm is used, the ECC is calculated while data are written to the memory. To read the correct ECC, the software must consequently wait until no write request to the NAND controller are pending, by polling PEF and NWRF bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWRF	Res.	PEF	Res.	Res.	ISOST[1:0]	
									r		r			r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **NWRF:** NAND write request flag

This bit provides the status of the write request issued to the NAND flash memory. When this bit is set, all write requests to the NAND controller are served.

0: NAND flash write requests are pending

1: No NAND flash write requests pending

Bit 5 Reserved, must be kept at reset value.

Bit 4 **PEF:** Pipe Empty Flag

This bit indicates if the pipe contains requests.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **ISOST[1:0]:** FMC isolation state with respect to the AXI interface

00: FMC is not isolated; AXI transactions are treated by the FMC and its controllers.

01: FMC has been enabled (FMCEN = 1) and waits for the end of pending AXI transactions (which leads to error responses) before going to “non-isolated” state.

10: FMC has been disabled (FMCEN = 0) and waits for the end of pending AXI transactions before going to isolation state.

11: FMC is isolated from its AXI interface. All AXI requests lead to error responses.

FMC common memory space timing register (FMC_PMEM)

Address offset: 0x88

Reset value: 0x0A0A 0A0A

The FMC_PMEM read/write register contains NAND flash memory region timing information. This information is used to access the NAND flash common memory space for command, address write accesses or data read/write accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEMHIZ[7:0]								MEMHOLD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMWAIT[7:0]								MEMSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **MEMHIZ[7:0]:** Common memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the data bus is kept Hi-Z after starting a NAND flash write access to the common memory space. This is only valid for write transactions:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycles

....

1111 1111: 256 * fmc_ker_ck cycles

Bits 23:16 **MEMHOLD[7:0]:** Common memory hold time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), during NAND flash read or write accesses to the common memory space:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycles

....

1111 1111: 256 * fmc_ker_ck cycles

Bits 15:8 **MEMWAIT[7:0]:** Common memory wait time

These bits define the minimum number of fmc_ker_ck clock cycles (+1) to assert the command (NWE, NOE), during NAND flash read or write accesses to the common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:

0000 0000: 1 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

0000 0001: 2 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

....

1111 1111: 256 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

Bits 7:0 **MEMSET[7:0]:** Common memory setup time

These bits define the number of fmc_ker_ck clock cycles (+1) to set up the address before the command assertion (NWE, NOE), during NAND flash read or write accesses to the common memory space:

0000 0000: 1 * fmc_ker_ck cycles

0000 0001: 2 * fmc_ker_ck cycles

....

1111 1111: 256 * fmc_ker_ck cycles

FMC attribute memory space timing registers (FMC_PATT)

Address offset: 0x8C

Reset value: 0x0A0A 0A0A

The FMC_PATT read/write register contains NAND flash memory region timing information. It is used for 8-bit accesses to the NAND flash attribute memory space during the last address write access when the timing differs from previous accesses (for Ready/Busy management, refer to *Section 25.9.5: NAND flash prewait function*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATTHIZ[7:0]								ATTHOLD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTWAIT[7:0]								ATTSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ATTHIZ[7:0]:** Attribute memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the data bus is kept in Hi-Z after starting a NAND flash write access to the attribute memory space. These bits are only valid for write transactions:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycle

....

1111 1111:256 * fmc_ker_ck cycles

Bits 23:16 **ATTHOLD[7:0]:** Attribute memory hold time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the address is held (and data for write access) after the command deassertion (NWE, NOE), during NAND flash read or write accesses to the attribute memory space:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycle

....

1111 1111:256 * fmc_ker_ck cycles.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

These bits define the minimum number of x fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), during NAND flash read or write accesses to the attribute memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:

0000 0000: 1 * fmc_ker_ck cycle (+ wait cycle introduced by deassertion of NWAIT)

0000 0001: 2 * fmc_ker_ck cycles (+ wait cycle introduced by deassertion of NWAIT)

....

1111 1111: 256 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

Bits 7:0 **ATTSET[7:0]:** Attribute memory setup time

These bits define the number of fmc_ker_ck clock cycles (+1) for address set up before the command assertion (NWE, NOE), during NAND flash read or write accesses to the attribute memory space:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycle

....

1111 1111:256 * fmc_ker_ck cycles.

FMC Hamming parity result registers (FMC_HPR)

Address offset: 0x90

Reset value: 0x0000 0000

This register is used during read accesses in conjunction with the FMC sequencer. It contains the current error correction code value computed by the FMC NAND controller Hamming module. When the FMC sequencer reads data from a NAND flash memory page at the correct address, the data read are automatically processed by the Hamming computation module. When X bytes have been read (according to the sector size ECCSS field in the FMC_PCR register), the CPU must read the computed ECC value from the FMC_HECCR register. It then verifies if these computed parity data are the same as the parity value recorded in the spare area and stored in the and the FMC_HPR, to determine whether a page is valid, and to correct it otherwise. The FMC_HPR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HPR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HPR[31:0]:** Hamming parity result

This field contains the parity value computed by the Hamming ECC computation module. *Table 173* describes the contents of these bitfields.

Table 173. HPR relevant bits

ECCSS[2:0]	Sector size in bytes	HPR bits
000	256	HPR[21:0]
001	512	HPR[23:0]
010	1024	HPR[25:0]
011	2048	HPR[27:0]
100	4096	HPR[29:0]
101	8192	HPR[31:0]

FMC Hamming code ECC result register (FMC_HECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the FMC NAND controller Hamming module.When the CPU reads/writes data from/to a NAND flash memory page at the correct address (refer to *Section 25.9.6: NAND ECC controller*), the data read/written from/to the NAND flash memory are automatically processed by the Hamming computation module. When X bytes have been read (according to the sector size ECCSS field in the FMC_PCR register), the CPU must read the computed ECC value from the FMC_HECCR register. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and to correct it otherwise. The FMC_HECCR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HECC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HECC[31:0]:** ECC result

This field contains the value computed by the Hamming ECC computation logic. *Table 174* describes the contents of these bitfields.

Table 174. ECC result relevant bits(1)

ECCSS[2:0]	Sector size in bytes	HECC bits
000	256	HECC[21:0]
001	512	HECC[23:0]
010	1024	HECC[25:0]
011	2048	HECC[27:0]
100	4096	HECC[29:0]
101	8192	HECC[31:0]

1. Only the relevant bits according to the sector size are valid.

FMC NAND interrupt enable register (FMC_IER)

Address offset: 0x180

Reset value: 0x0000 0000

This register configures the interrupt capability generation when the RNB signal is connected on FMC interrupt pin. An interrupt can be generated on the rising edge, falling edge or high level detected on the RNB signal.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEE	IHLE	IREE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IFEE:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 1 **IHLE:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 0 **IREE:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

FMC interrupt status register (FMC_ISR)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEF	IHLF	IREF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IFEF:** Interrupt falling edge flag
The flag is set by hardware and cleared by setting the CFEIF bit.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

Bit 1 **IHLF:** Interrupt high-level flag
The flag is set by hardware and cleared by setting the CHLIF bit.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IREF:** Interrupt rising edge flag
The flag is set by hardware and cleared by setting the CREIF bit.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

FMC NAND controller interrupt clear register (FMC_ICR)

Address offset: 0x188

Reset value: 0x0000 0000

Writing 1 in one of these bits clears the corresponding flag in the interrupt status register (FMC_ISR).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIFEF	CIHLF	CIREF
													w	w	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CIFEF:** Clear Interrupt falling edge flag
0: No effect
1: Clears the IFEF flag in the FMC_ISR register

Bit 1 **CIHLF:** Clear Interrupt high-level flag
0: No effect
1: Clears the IHLF flag in the FMC_ISR register

Bit 0 **CIREF:** Clear Interrupt rising edge flag
0: No effect
1:Clear the IREF flag in the FMC_ISR register

FMC NAND command sequencer control register (FMC_CSQCR)

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSQSTART
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **CSQSTART**: Command Sequencer Enable
Writing 1 starts the Command sequencer.
Writing 0 has no effect.

Note: Write this bit to 1 to start the sequencer only when the FMC is enabled by setting FMCEN bit of FMC_CFGR to 1 and the NAND flash controller is activated by setting PBKEN bit of FMC_PCR to 1.

FMC NAND command sequencer configuration register 1 (FMC_CSQCFGR1)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CMD2T	CMD1T	CMD2[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD1[7:0]								Res.	ACYNBR[2:0]			Res.	DMADEN	CMD2EN	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **CMD2T:** Command 2 Sequencer timings

This bit specifies if the Command 2 (CMD2) is issued to the NAND flash memory with the timings defined in FMC_PMEM or FMC_PATT register.

0: CMD2 issued with the timings programmed in FMC_PMEM

1: CMD2 issued with the timings programmed in FMC_PATT

Bit 24 **CMD1T:** Command 1 Sequencer timings

This bit specifies if the Command 1 (CMD1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bits 23:16 **CMD2[7:0]:** Command 2 sequencer

These bits specify if the Command 2 is issued by the command sequencer for the command second cycle during read operations.This command is only issued when the CMD2EN bit is set.

Example: CMD2[7:0] bits must be programmed to 0x30 to issue the read column command.

Bits 15:8 **CMD1[7:0]:** Command 1 sequencer

These bits specify if the Command 1 is issued by the command sequencer for the first cycle during read or write operations.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACYNBR[2:0]:** Address Cycle number

These bits define the number of address cycles to be generated by the command sequencer when issuing the command programmed in the registers.

000: No address cycle.

001: 1 address cycle

010: 2 address cycles

011: 3 address cycles

100: 4 address cycles

101: 5 address cycles

Others: Reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DMADEN:** Command sequencer DMA request data enable

When this bit is set, the command sequencer generates a DMA request to the first DMA channel, to transfer data from/to NAND flash memory.

0: No DMA request transfer

1: A DMA request transfer

Bit 1 **CMD2EN:** Command cycle 2 Enable

This bit specifies that the command cycle 2 has to be generated by the command sequencer.

0: Command cycle 2 not issued.

1: Command cycle 2 (programmed CMD2[7:0]) sent by the command sequencer to the NAND flash memory after the address cycles.

Bit 0 Reserved, must be kept at reset value.

FMC NAND command sequencer configuration register 2 (FMC_CSQCFGR2)

Address offset:0x208

Reset value: 0x0000 0000

This register is used to configure the command sequencer to issue random read/ write commands to read/ write data by sector and automatically read/write data from NAND flash memory at a programmable address offset. This is useful when performing a sector read/write operation followed by an ECC read/write operation in the NAND flash spare area.The command sequencer generates the random commands until all the sectors are read/written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RCMD2T	RCMD1T	RCMD2[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCMD1[7:0]								Res.	Res.	Res.	Res.	Res.	DMASEN	RCMD2EN	SQSDTEN
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RCMD2T:** Command 1 sequencer timings

This bit specifies if the CMD1 is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bit 24 **RCMD1T:** Command 1 sequencer timings

This bit specifies if the CMD1 is issued to the NAND flash memory with the timings flash memory in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bits 23:16 **RCMD2[7:0]:** Random Command 2 sequencer

These bits specify that the command 2 is issued by the command sequencer for the second cycle during read operations.This command is only issued when the RCMD2EN bit is set.

Example: the RCMD2 (0xE0) is required to issue the random read column command.

Bits 15:8 **RCMD1[7:0]:** Random Command 1 sequencer

These bits specify that the command 1 value is issued by the command sequencer for the first cycle during read or write accesses.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **DMASEN:** Command sequencer DMA request decoding status enable

When this bit is set, the command sequencer generates a DMA request to the second DMA channel when valid bits are available in the ECC status register at the end of ECC processing (decoding).

0: No DMA request used for ECC status registers transfer

1: A DMA request used for ECC status registers transfer

Note: This bit must be set only during a page read sequence to offload the saving of the ECC results between sectors by using a second DMA channel.

Bit 1 **RCMD2EN:** Random Command 2 sequencer enable

This bit enables the command 2 to be issued by the command sequencer during read accesses. This command is issued only when the CMD2SQEN bit is set.

0: Command 2 not issued.

1: Command 2 (CMD2SQ[7:0]) issued by the command sequencer to NAND flash memory after the address cycle.

Bit 0 **SQSDTEN:** Sequencer spare data transfer enable

This bit enables the sequencer to access the spare data area after each sector transfer. It also enables the ECC when using the sequencer. ECCEN bit in FMC_PCR register must be reset.

0: ECC disabled and spare data area not accessed by the sequencer

1: ECC enabled and spare data area read or programmed by the sequencer after each sector transfer

FMC NAND sequencer configuration register 3 (FMC_CSQCFGR3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAC2T	RAC1T	SDT	AC5T	AC4T	AC3T	AC2T	AC1T
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	SNBR[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **RAC2T:** Random Address cycle 2 sequencer timings

This bit specifies if the random Address cycle 2 (ADDC2) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or in FMC_PATT register.

0: Random ADDC2 issued with the timings programmed in FMC_PMEM

1: Random ADDC2 issued with the timings programmed in FMC_PATT

Bit 22 **RAC1T:** Random Address cycle 1 sequencer timings

This bit specifies if the random Address cycle 1 (ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or in FMC_PATT register.

0: Random ADDC1 issued with the timings programmed in FMC_PMEM

1: Random ADDC1 issued with the timings programmed in FMC_PATT

Bit 21 **SDT:** Spare data transfer sequencer timings

This bit specifies if the spare data transfer for read and write operations from/to the NAND flash memory are performed with the timings programmed in FMC_PMEM or FMC_PATT register.

0: Spare data transfer issued with the timings programmed in FMC_PMEM

1: Spare data transfer issued with the timings programmed in FMC_PATT

Bit 20 **AC5T:** Address cycle 5 sequencer timings

This bit defines if the Address cycle 5(ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC5 issued with the timings programmed in FMC_PMEM

1: ADDC5 issued with the timings programmed in FMC_PATT

Bit 19 **AC4T:** Address cycle 4sequencer timings

This bit defines if the Address cycle 4 (ADDC4) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC4 issued with the timings programmed in FMC_PMEM

1: ADDC4 issued with the timings programmed in FMC_PATT

Bit 18 **AC3T:** Address cycle 3 sequencer timings

This bit defines if the Address cycle 3 (ADDC3) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC3 issued with the timings programmed in FMC_PMEM

1: ADDC3 issued with the timings programmed in FMC_PATT

Bit 17 **AC2T:** Address cycle 2 sequencer timings

This bit defines if the Address cycle 2 (ADDC2) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC2 issued with the timings programmed in FMC_PMEM

1: ADDC2 issued with the timings programmed in FMC_PATT

Bit 16 **AC1T:** Address cycle 1 sequencer timings

This bit defines if the Address cycle 1 (ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC1 issued with the timings programmed in FMC_PMEM

1: ADDC1 issued with the timings programmed in FMC_PATT

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **SNBR[5:0]:** Number of sectors to be read/written

These bits define the number of sectors that are processed by the sequencer.

They are used to read/write part of a page or to use the ECC (in particular with the BCH code which requires 512-byte sectors):

000000: 1 sector

000001: 2 sectors

...

111111: 16 sectors

Others: Reserved.

Bits 7:0 Reserved, must be kept at reset value.

FMC NAND command sequencer address register 1 (FMC_CSQAR1)

Address offset: 0x210

Reset value: 0x0000 0000

This register is used to define the value of address cycles 1 to 4 to be issued by the command sequencer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDC4[7:0]								ADDC3[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDC2[7:0]								ADDC1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ADDC4[7:0]:** Address Cycle 4

These bits define the value of address cycle 4 to be issued by the command sequencer during read or write accesses.

Bits 23:16 **ADDC3[7:0]:** Address Cycle 3

These bits define the value of address cycle 2 to be issued by the command sequencer during read or write accesses.

Bits 15:8 **ADDC2[7:0]:** Address Cycle 2

These bits define the value of address cycle 2 to be issued by the command sequencer during read or write accesses.

Bits 7:0 **ADDC1[7:0]:** Address Cycle 1

These bits define the value of address cycle 1 to be issued by the command sequencer during read or write accesses.

FMC NAND command sequencer address register 2 (FMC_CSQAR2)

Address offset: 0x214

Reset value: 0x0000 0000

This register is used to program the fifth address cycle and the address offset in spare area. It also selects the chip enable.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NAND CEN1	NAND CEN0	Res.	Res.	ADDC5[7:0]							
				rw	rw			rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **SAO[15:0]:** Spare Area Address Offset

These bits define the offset of the first ECC byte in the spare area for the program sequence.

Note: The minimum SAQ value is 2 since the first two bytes of the spare area are reserved for Bad block marking and metadata.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:10 **NANDCEN[1:0]:** NAND flash chip enable number

These bits define the chip enable to be selected by the NAND flash command sequencer.

00: NCE1 chip enable selected

01: NCE2 chip enable selected

Others: Reserved

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:0 **ADDC5[7:0]:** Address Cycle 5

These bits define the value of address cycle 5 to be issued by the command sequencer during read or write accesses.

FMC NAND command sequencer interrupt enable register (FMC_CSQIER)

Address offset: 0x220

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCIE	SUEIE	SEIE	SCIE	TCIE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDTCIE:** Command Transfer Complete interrupt enable
0: Command Transfer Complete Interrupt disable
1: Command Transfer Complete Interrupt enable

Bit 3 **SUEIE:** Sector Uncorrectable Error interrupt enable
0: Command Transfer Complete Interrupt disable
1: Command Transfer Complete Interrupt enable

Bit 2 **SEIE:** Sector Error interrupt enable
0: Sector Error Interrupt disable
1: Sector Error Interrupt enable

Bit 1 **SCIE:** Sector Complete interrupt enable
0: Sector Complete Interrupt disable
1: Sector Complete Interrupt enable

Bit 0 **TCIE:** Transfer Complete Interrupt enable
0: Transfer Complete Interrupt disable
1: Transfer Complete Interrupt enable

FMC NAND command sequencer interrupt status register (FMC_CSQISR)

Address offset: 0x224

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCF	SUEF	SEF	SCF	TCF
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CMDTCF:** Command Transfer Complete flag

This bit is set when the command sequencer has completed the transfer of programmed commands and addresses.

This bit is cleared by setting CCMDTCF bit. An interrupt can be generated if the CMDTCIE bit is set in the FMC_CSQICR register.

Note: The Sector Transfer Complete interrupt is not needed in sequencer mode.

Bit 3 **SUEF:** Sector Uncorrectable Error flag

This bit is set by hardware and cleared by writing 1 to the CSUEF bit in FMC_CSQICR.

When this bit is set, it indicates that the command sequencer detected an uncorrectable error when decoding a sector. An interrupt can be generated if the SUEIE bit is set in the FMC_CSQICR register.

Note: The SUEF bit is relevant only for BCH code to indicate uncorrectable errors. It must be ignored for Hamming code.

Bit 2 **SEF:** Sector Error flag
This bit is set by hardware and cleared by writing 1 to the CSEF bit in FMC_CSQICR.
This bit is set when the command sequencer has completed the sector data transfer with at least one error detection. An interrupt can be generated if the SEIE bit is set in the FMC_CSQICR register.
Note: The SEF bit is relevant only for BCH code to indicate that errors are present. It must be ignored for Hamming code.

Bit 1 **SCF:** Sector Complete flag
This bit is set by hardware and cleared by writing 1 to the CSCF bit in FMC_CSQICR.
This bit is set when the command sequencer has completed the sector data transfer. An interrupt can be generated if the SCIE bit is set in the FMC_CSQICR register.

Bit 0 **TCF:** Transfer Complete flag
This bit is set by hardware and cleared by writing 1 to the CTCF bit in FMC_CSQICR.
This bit is set when the command sequencer has transferred all the data for all sectors or when the command sequencer was aborted. An interrupt can be generated if the TCIE bit is set in the FMC_CSQICR register.

FMC NAND command sequencer interrupt clear register (FMC_CSQICR)

Address offset: 0x228

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDTCF	CSUEF	CSEF	CSCF	CTCF
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CCMDTCF:** Clear Command Transfer Complete flag
Writing one clears the CMTCF flag in the FMC_CSQISR register

Bit 3 **CSUEF:** Clear Sector uncorrectable Error flag
Writing one clears the SUEF flag in the FMC_CSQISR register

Bit 2 **CSEF:** Clear Sector Error flag
Writing one clears the SEF flag in the FMC_CSQISR register

Bit 1 **CSCF:** Clear Sector Complete flag
Writing one clears the SCF flag in the FMC_CSQISR register

Bit 0 **CTCF**: Clear Transfer Complete flag
Writing one clears the TCF flag in the FMC_CSQISR register

FMC command sequencer error mapping status register (FMC_CSQEMSR)

Address offset: 0x230

Reset value: 0x0000 0000

This register holds a sector error mapping status when the whole transfer is complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEM[15:0]:** Sector Error mapping
When SEM[i] bit (0 ≤ i ≤ 15) is set, at least one error has been detected on sector i+1.
As an example, SEM[2] and SEM[1] bits are set when errors are detected in sector 3 and 2.

FMC BCH interrupt enable register (FMC_BCHIER)

Address offset: 0x250

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRIE	DSRIE	DEFIE	DERIE	DUEIE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EPBRIE:** Decoder Parity Bits Ready Interrupt enable
0: Decoder Parity Bits Ready Interrupt disable
1: Decoder Parity Bits Ready Interrupt enable

Bit 3 **DSRIE:** Decoder Syndrome Ready Interrupt enable
0: Decoder Syndrome Ready Interrupt disable
1: Decoder Syndrome Ready Interrupt enable

Bit 2 **DEFIE:** Decoder Error Found Interrupt enable
0: Decoder Error Found Interrupt disable
1: Decoder Error Found Interrupt enable

Bit 1 **DERIE:** Decoder Error Ready Interrupt enable
0: Decoder Error Ready Interrupt disable
1: Decoder Error Ready Interrupt enable

Bit 0 **DUEIE:** Decoder Uncorrectable Errors Interrupt enable
0: Decoder Uncorrectable Errors Interrupt disable
1: Decoder Uncorrectable Errors Interrupt enable

FMC BCH interrupt and status register (FMC_BCHISR)

Address offset: 0x254

Reset value: 0x0000 0000

This register holds the status of BCH encoder/decoder after processing each sector. When the sequencer is used, this register is automatically cleared and the corresponding interrupt must not be enabled through FMC_BCHIER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRF	DSRF	DEFF	DERF	DUEF
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EPBRF:** Encoder Parity Bits Ready flag
This bit is set when the BCH encoder parity bits are available in FMC_BCHPBR1 to 4.

Bit 3 **DSRF:** Decoder Syndrome Ready flag
This bit is set when the syndrome is ready and DEFF is a valid field.

Bit 2 **DEFF:** Decoder Error Found flag
This bit is set by hardware and cleared by writing 1 to CDEFF.
This bit indicates that at least one error has been detected in the sector.

Bit 1 **DERF:** Decoder Error Ready flag
This bit indicates that the decoder has finished searching errors and identifying their location. The number of errors and their location are available in FMC_BCHDSR0 and FMC_BCHDSR1 to 4, respectively.

Bit 0 **DUEF:** Decoder Uncorrectable Errors flag
This field indicates that too many errors have been detected in the sector during the BCH decoding and that the sector is uncorrectable.

FMC BCH interrupt clear register (FMC_BCHICR)

Address offset: 0x258

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEPBRF	CDSRF	CDEFF	CDERF	CDUEF
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CEPBRF:** Clear Encoder Parity Bits Ready flag
Writing 1 clears the EPBRF flag in FMC_BCHISR.

Bit 3 **CDSRF:** Clear Decoder Syndrome Ready flag
Writing 1 clears the DSRF flag in FMC_BCHISR.

Bit 2 **CDEFF:** Clear Decoder Error Found flag
Writing 1 clears the DEFF flag in FMC_BCHISR.

Bit 1 **CDERF:** Clear Decoder Error ready flag
Writing 1 clears the DERF flag in FMC_BCHISR.

Bit 0 **CDUEF:** Clear Decoder Uncorrectable Error flag
Writing 1 clears the DUEF flag in FMC_BCHISR.

FMC BCH parity bits register 1 (FMC_BCHPBR1)

Address offset: 0x260

Reset value: 0x0000 0000

These registers contain the BCH parity bits (BCHPB). For the BCH 4-bit, only BCHPB[51:0] are significant and for the BCH 8-bit BCHPB[103:0] are significant.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BCHPB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[31:0]:** BCH parity bits

FMC BCH parity bits register 2 (FMC_BCHPBR2)

Address offset: 0x264

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[63:32]:** BCH parity bits

FMC BCH parity bits register 3 (FMC_BCHPBR3)

Address offset: 0x268

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[95:64]:** BCH parity bits

FMC BCH parity bits register 4 (FMC_BCHPBR4)

Address offset: 0x26C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCHPB[103:96]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **BCHPB[103:96]:** BCH parity bits

FMC BCH decoder status register 0 (FMC_BCHDSR0)

Address offset: 0x27C

Reset value: 0x0000 0000

This register contains some fields already available in other registers but that require to be saved when error correction is performed on several sectors at a time (for example a whole NAND flash page). This allows a DMA channel to transfer the content of FMC_BCHDSR0 to 4 to a decoding status buffer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN[3:0]				Res.	Res.	DEF	DUE
								r	r	r	r			r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DEN[3:0]:** Decoder error number

When DEF bit is set, this field indicates the number of errors detected by the BCH decoder. Following the selected BCH code, if the number of detected errors exceeds the number of correctable errors, this bit is not significant and DUE bit is set.

If the BCH decoder detects a number of errors that exceeds the maximum number of detectable errors, DEN is not significant.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DEF:** Decoder error found

This field indicates that the decoder has finished searching errors and identifying their location.

The number of errors and their location are available in FMC_BCHDSR0 and FMC_BCHDSR1 to 4, respectively.

Bit 0 **DUE:** Decoder uncorrectable error

This field indicates that too many errors have been detected in the sector during the BCH decoding and that the sector is uncorrectable.

If the BCH decoder detects a number of errors that exceeds the maximum number of detectable errors, DUE is not significant.

FMC BCH decoder status register for memory region 1 (FMC_BCHDSR1)

Address offset: 0x280

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 1st and 2nd error bits in EBP1 and EPB2 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are “don’t care”.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP2[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP1[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP2[12:0]**: Error bit position for error number 2

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP1[12:0]**: Error bit position for error number 1

FMC BCH decoder status register for memory region 2 (FMC_BCHDSR2)

Address offset: 0x284

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 3rd and 4th error bits in EBP3 and EPB4 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don’t care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP4[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP3[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP4[12:0]**: Error bit position for error number 4

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP3[12:0]**: Error bit position for error number 3

FMC BCH decoder status register for memory region 3 (FMC_BCHDSR3)

Address offset: 0x288

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 5th and 6th error bits in EBP5 and EPB6 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don’t care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP6[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	EBP5[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP6[12:0]**: Error bit position for error number 6

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP5[12:0]**: Error bit position for error number 5

FMC BCH decoder status register for memory region 4 (FMC_BCHDSR4)

Address offset: 0x28C

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 7th and 8th error bits in EBP7 and EPB8 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don’t care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP8[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	EBP7[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP8[12:0]**: Error bit position for error number 8

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP7[12:0]**: Error bit position for error number 7

25.10 FMC register map

Table 175. FMC register map

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMC_BCR1	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value	0								0	0		0	0	0	0	0	0	0	0	0	0		1	1		1	0	1	1	0	1	1
0x08	FMC_BCR2	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value	0								0	0		0	0	0	0	0	0	0	0	0	0		1	1		1	0	1	0	0	1	0
0x10	FMC_BCR3	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value	0								0	0		0	0	0	0	0	0	0	0	0	0		1	1		1	0	1	0	0	1	0
0x18	FMC_BCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		Res.	CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value	0								0	0		0	0	0	0	0	0	0	0	0	0		1	1		1	0	1	0	0	1	0
0x04	FMC_BTR1	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	FMC_BTR2	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	FMC_BTR3	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	FMC_BTR4	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x20	FMC_PCSCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTB4EN	CNTB3EN	CNTB2EN	CNTB1EN	CSCOUNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	FMC_PCR	Res.	Res.	Res.	Res.	Res.	Res.	WEN	BCHECC	Res.	Res.	Res.	Res.	ECCSS[2:0]			TAR[3:0]				TCLR[3:0]				ECCALG	Res.	ECCEN	PWID		Res.	PBKEN	PWAITEN	Res.
	Reset value							0	0					0	1	1	1	1	1	1	1	1	1	1	0		0	0	0		0	0	
0x84	FMC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWRF	Res.	PEF	Res.	Res.	ISOST	
	Reset value																										1		1			1	1
0x88	FMC_PMEM	MEMHIZ								MEMHOLD								MEMWAIT								MEMSET							
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0
0x8C	FMC_PATT	ATTHIZ								ATTHOLD								ATTWAIT								ATTSET							
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0

Table 175. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x90	FMC_HPR	HPR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	FMC_HECCR	HECC[[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	Reserved	Reserved																															
0x104	FMC_BWTR1	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x10C	FMC_BWTR2	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	FMC_BWTR3	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x11C	FMC_BWTR4	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x180	FMC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEE	IHLE	IREE
	Reset value																														0	0	0
0x184	FMC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEF	IHLF	IREF
	Reset value																														0	0	0
0x188	FMC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIFEF	CIHLF	CIREF
	Reset value																														0	0	0
0x200	FMC_CSQCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSQSTART
	Reset value																																0
0x204	FMC_ CSQCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	CMD2T	CMD1T	CMD2								CMD1								ACYNBR			Res.	Res.	DMADEN	CMD2EN	Res.
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	
0x208	FMC_ CSQCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	RCMD2T	RCMD1T	RCMD2								RCMD1								Res.	Res.	Res.	Res.	Res.	DMADEN	RCMD2EN	SQSDTEN
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x20C	FMC_ CSQCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAC2T	RAC1T	SDT	AC5T	AC4T	AC3T	AC2T	AC1T	Res.	Res.	SNBR						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0			0	0	0	0	0	0								
0x210	FMC_CSQAR1	ADDCA4								ADDC3								ADDC2								ADDC1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x214	FMC_CSQAR2	SAO																Res.	Res.	Res.	Res.	NANDCEN1	NANDCEN0	Res.	Res.	ADDC5							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0					0	0			0	0	0	0	0	0	0	0

Table 175. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x220	FMC_CSQIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCIE	SUEIE	SEIE	SCIE	TCIE
	Reset value																												0	0	0	0	0
0x224	FMC_CSQISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCF	SUEF	SEF	SCF	TCF
	Reset value																												0	0	0	0	0
0x228	FMC_CSQICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDTCF	CSUEF	CSEF	CSCF	CTCF
	Reset value																												0	0	0	0	0
0x22C	Reserved	Reserved																															
0x230	FMC_CSQEMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x240	Reserved	Reserved																															
0x250	FMC_BCHIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRIE	DSRIE	DEFIE	DERIE	DUEIE
	Reset value																												0	0	0	0	0
0x254	FMC_BCHISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRF	DSRF	DEFF	DERF	DUEF
	Reset value																												0	0	0	0	0
0x258	FMC_BCHICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEPBRF	CDSRF	CDEFF	CDERF	CDUEF
	Reset value																												0	0	0	0	0
0x260	FMC_BCHPBR1	BCHPB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x264	FMC_BCHPBR2	BCHPB[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x268	FMC_BCHPBR3	BCHPB[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x26C	FMC_BCHPBR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCHPB[103:96]							
	Reset value																									0	0	0	0	0	0	0	0
0x27C	FMC_BCHDSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN				Res.	Res.	DEF	DUE
	Reset value																									0	0	0	0			0	0
0x280	FMC_BCHDSR1	Res.	Res.	Res.	EBP2[12:0]													Res.	Res.	Res.	EBP1[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x284	FMC_BCHDSR2	Res.	Res.	Res.	EBP4[12:0]													Res.	Res.	Res.	EBP3[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x288	FMC_BCHDSR3	Res.	Res.	Res.	EBP6[12:0]													Res.	Res.	Res.	EBP5[12:0]												
	Reset value																																
0x28C	FMC_BCHDSR4	Res.	Res.	Res.	EBP8[12:0]													Res.	Res.	Res.	EBP7[12:0]												
	Reset value																																

Table 175. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3EC	FMC_HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDRAM2_BASE [3:0]				SDRAM1_BASE[3:0]				NAND_BASE [3:0]				SDRAM_RBASE [3:0]				NOR_BASE [3:0]				RD_LN2PTH [3:0]			
	Reset value	0x00DC 8762																															
0x3F0	FMC_HWCFGR1	RA_LN2DPTH [3:0]				WR_LN2DPTH [3:0]				WD_LN2DPTH [3:0]				WA_LN2DPTH [3:0]				ID_SIZE[3:0]				Res.	Res.	Res.	SDRAM_SEL	Res.	Res.	Res.	NAND_ECC	Res.	Res.	Res.	NAND_SEL
	Reset value	0x2232 B011																															
0x3F4	FMC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0011																															
0x3F8	FMC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	FMC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

26 Quad-SPI interface (QUADSPI)

26.1 Introduction

The QUADSPI is a specialized communication interface targeting single, dual- or quad-SPI flash memories. It can operate in any of the three following modes:

- indirect mode: all the operations are performed using the QUADSPI registers.
- automatic status-polling mode: the external flash memory status register is periodically read and an interrupt can be generated in case of flag setting.
- memory-mapped mode: the external flash memory is mapped to the device address space and is seen by the system as if it was an internal memory.

Both throughput and capacity can be increased two-fold using dual-flash mode, where two Quad-SPI flash memories are accessed simultaneously.

26.2 QUADSPI main features

- Three functional modes: indirect, automatic status-polling, and memory-mapped
- Dual-flash mode, where 8 bits can be sent/received simultaneously by accessing two flash memories in parallel
- SDR and DDR support
- Fully programmable opcode for both indirect and memory-mapped modes
- Fully programmable frame format for both indirect and memory-mapped modes
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error

26.3 QUADSPI functional description

26.3.1 QUADSPI block diagram

Figure 168. QUADSPI block diagram

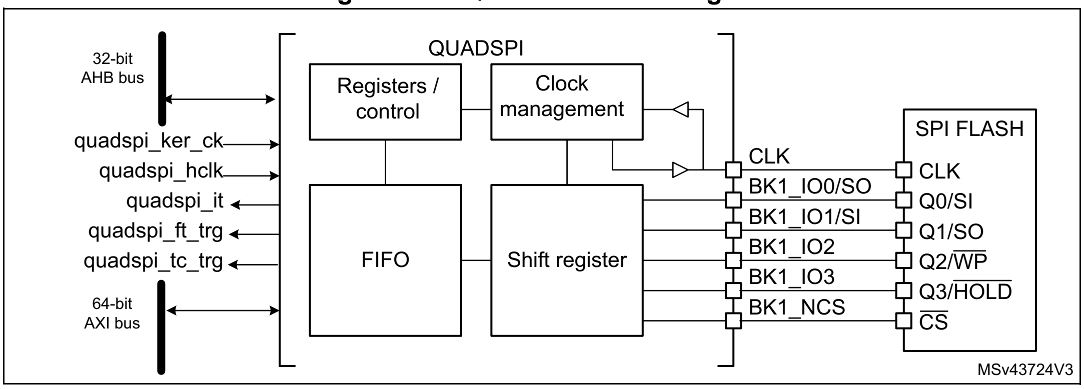
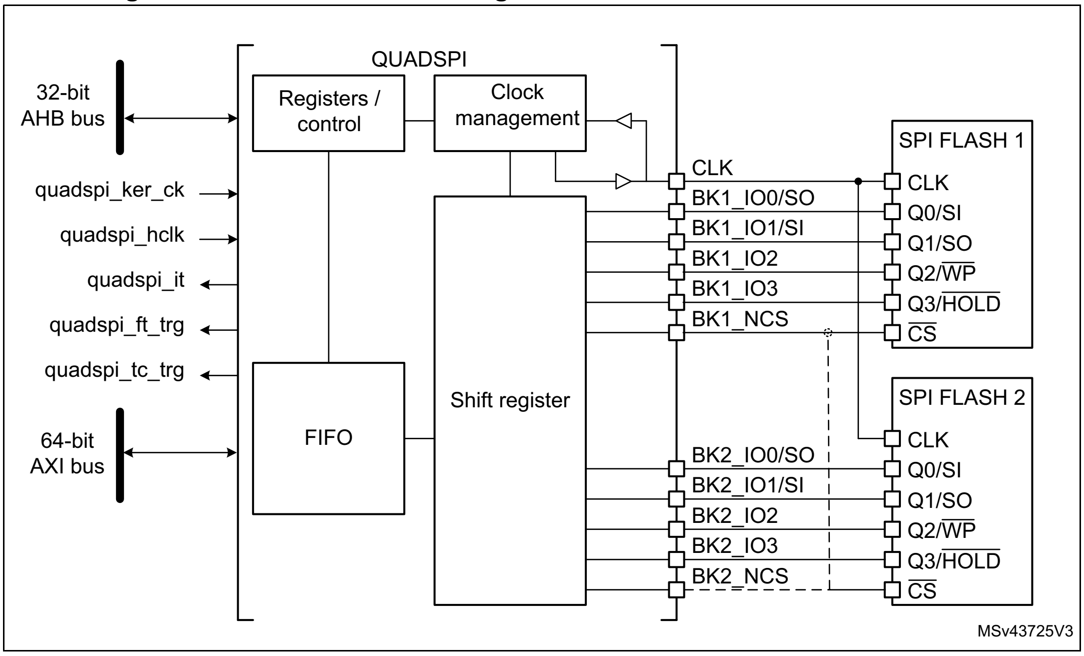


Figure 169. QUADSPI block diagram when dual-flash mode is enabled



26.3.2 QUADSPI pins and internal signals

The table below lists the QUADSPI internal signals.

Table 176. QUADSPI internal signals

Signal name	Signal type	Description
quadspi_ker_ck	Digital input	QUADSPI kernel clock
quadspi_hclk	Digital input	QUADSPI register interface clock
quadspi_it	Digital output	QUADSPI global interrupt
quadspi_ft_trg	Digital output	QUADSPI FIFO threshold trigger for MDMA
quadspi_tc_trg	Digital output	QUADSPI transfer complete trigger for MDMA

The table below lists the QUADSPI pins, six for interfacing with a single flash memory, or 10 to 11 for interfacing with two flash memories (FLASH 1 and FLASH 2) in dual-flash mode.

Table 177. QUADSPI pins

Signal name	Signal type	Description
CLK	Digital output	Clock to FLASH 1 and FLASH 2
BK1_IO0/SO	Digital input/output	Bidirectional I/O in dual/quad modes or serial output in single mode, for FLASH 1
BK1_IO1/SI	Digital input/output	Bidirectional I/O in dual/quad modes or serial input in single mode, for FLASH 1
BK1_IO2	Digital input/output	Bidirectional I/O in quad mode, for FLASH 1
BK1_IO3	Digital input/output	Bidirectional I/O in quad mode, for FLASH 1
BK2_IO0/SO	Digital input/output	Bidirectional I/O in dual/quad modes or serial output in single mode, for FLASH 2

Table 177. QUADSPI pins (continued)

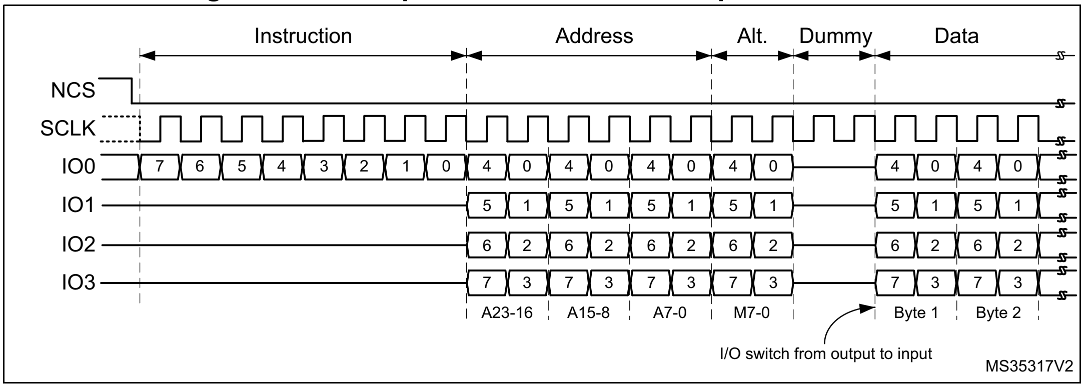
Signal name	Signal type	Description
BK2_IO1/SI	Digital input/output	Bidirectional I/O in dual/quad modes or serial input in single mode, for FLASH 2
BK2_IO2	Digital input/output	Bidirectional I/O in quad mode, for FLASH 2
BK2_IO3	Digital input/output	Bidirectional I/O in quad mode, for FLASH 2
BK1_NCS	Digital output	Chip select (active low) for FLASH 1. Can also be used for FLASH 2 if QUADSPI is always used in dual-flash mode.
BK2_NCS	Digital output	Chip select (active low) for FLASH 2. Can also be used for FLASH 1 if QUADSPI is always used in dual-flash mode.

26.3.3 QUADSPI command sequence

The QUADSPI communicates with the flash memory using commands. Each command can include five phases: instruction, address, alternate byte, dummy, data. Any of these phases can be configured to be skipped, but at least one of the instruction, address, alternate byte, or data phase must be present.

NCS falls before the start of each command and rises again after each command finishes.

Figure 170. Example of read command in quad-SPI mode



Instruction phase

During this phase, an 8-bit instruction, configured in INSTRUCTION bitfield of QUADSPI_CCR[7:0] register, is sent to the flash memory, specifying the type of operation to be performed.

Most flash memories can receive instructions only one bit at a time from the IO0/SO signal (single-SPI mode), the instruction phase can optionally send 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the IMODE[1:0] bitfield of QUADSPI_CCR[9:8] register.

When IMODE = 00, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

Address phase

In the address phase, 1-4 bytes are sent to the flash memory to indicate the address of the operation. The number of address bytes to be sent is configured in the ADSIZE[1:0] bitfield of QUADSPI_CCR[13:12] register. In indirect and automatic status-polling modes, address bytes to be sent are specified in the ADDRESS[31:0] bitfield of QUADSPI_AR register, while in memory-mapped mode, the address is given directly via the AHB (from the Cortex or from a DMA).

The address phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ADMODE[1:0] bitfield of QUADSPI_CCR[11:10] register.

When ADMODE = 00, the address phase is skipped, and the command sequence proceeds directly to the next phase, if any.

Alternate-byte phase

In the alternate-byte phase, 1-4 bytes are sent to the flash memory, generally to control the mode of operation. The number of alternate bytes to be sent is configured in the [1:0] bitfield of QUADSPI_CCR[17:16] register. The bytes to be sent are specified in the QUADSPI_ABR register.

The alternate-bytes phase can send 1 bit at a time (over SO in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] bitfield of QUADSPI_CCR[15:14] register.

When ABMODE = 00, the alternate-byte phase is skipped, and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-mode is used and only two cycles are used for the alternate bytes. In this case, the firmware can use quad-mode (ABMODE = 11) and send a byte with bits 7 and 3 of ALTERNATE set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low). In this case, the upper two bits of the nibble to be sent are placed in bits 4:3 of ALTERNATE, while the lower two bits are placed in bits 1 and 0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE must be set to 0x8A (1000_1010).

Dummy-cycle phase

In the dummy-cycle phase, 1-31 cycles are given without any data being sent or received, in order to give time to the flash memory to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in the DCYC[4:0] bitfield of QUADSPI_CCR[22:18] register. In both SDR and DDR modes, the duration is specified as a number of full CLK cycles.

When DCYC is zero, the dummy-cycles phase is skipped, and the command sequence proceeds directly to the data phase, if present.

The operating mode of the dummy-cycles phase is determined by DMODE.

In order to assure enough “turn-around” time for changing data signals from output mode to input mode, there must be at least one dummy cycle when using dual or quad mode to receive data from the flash memory.

Data phase

During the data phase, any number of bytes can be sent to, or received from the flash memory.

In indirect and automatic status-polling modes, the number of bytes to be sent/received is specified in the QUADSPI_DLR register.

In indirect-write mode the data to be sent to the flash memory must be written to the QUADSPI_DR register. In indirect-read mode the data received from the flash memory is obtained by reading the QUADSPI_DR register.

In memory-mapped mode, the data which is read is sent back directly over the AHB to the Cortex or to a DMA.

The data phase can send/receive 1 bit at a time (over SO/SI in single-SPI mode), 2 bits at a time (over IO0/IO1 in dual-SPI mode), or 4 bits at a time (over IO0/IO1/IO2/IO3 in quad-SPI mode). This can be configured using the ABMODE[1:0] bitfield of QUADSPI_CCR[15:14] register.

When DMODE = 00, the data phase is skipped, and the command sequence finishes immediately by raising NCS. This configuration must only be used in only indirect write mode.

26.3.4 QUADSPI signal interface protocol modes

Single-SPI mode

This legacy SPI mode allows just one single bit to be sent/received serially. In this mode, data are sent to the flash memory over the SO signal (whose I/O shared with IO0). Data received from the flash memory arrive via SI (whose I/O shared with IO1).

The different phases can each be configured separately to use this mode by setting the IMODE/ADMODE/ABMODE/DMODE fields (in QUADSPI_CCR) to 01.

In each phase which is configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).

This is the case even for the dummy phase if DMODE = 01.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 10.

In each phase which is configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.

In the dummy phase when DMODE = 01, IO0/IO1 are always high-impedance.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use quad-SPI mode by setting the IMODE/ADMODE/ABMODE/DMODE fields of QUADSPI_CCR register to 11.

In each phase which is configured in this mode, IO0/IO1/IO2/IO3 are all are at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE = 11, IO0/IO1/IO2/IO3 are all high-impedance.

IO2 and IO3 are used only in quad-SPI mode. If none of the phases are configured to use quad-SPI mode, then the pins corresponding to IO2 and IO3 can be used for other functions even while the QUADSPI is active.

SDR mode

By default, the DDRM bit (QUADSPI_CCR[31]) is 0 and the QUADSPI operates in single-data rate (SDR) mode.

In SDR mode, when the QUADSPI drives IO0/SO, IO1, IO2, IO3 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the QUADSPI assumes that flash memories also send the data using CLK falling edge. By default (when SSHIFT = 0), the signals are sampled using the following (rising) edge of CLK.

DDR mode

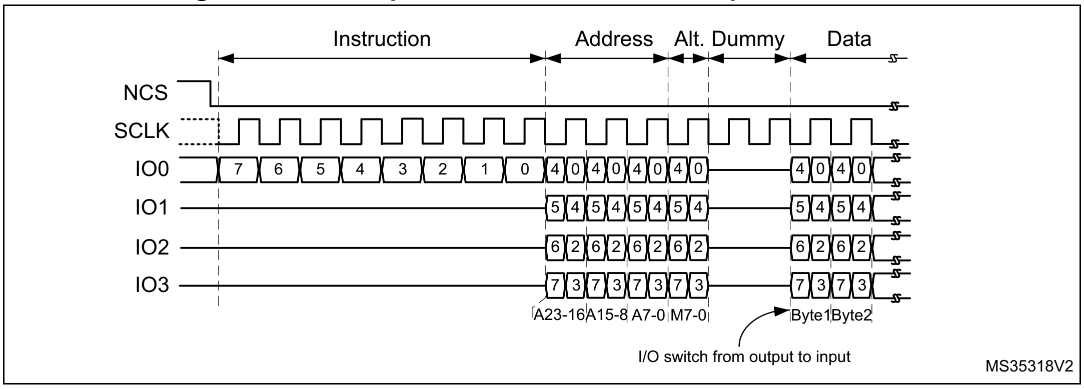
When the DDRM bit (QUADSPI_CCR[31]) is set to 1, the QUADSPI operates in double-data rate (DDR) mode.

In DDR mode, when the QUADSPI is driving the IO0/SO, IO1, IO2, IO3 signals in the address/alternate-byte/data phases, a bit is sent on each of CLK falling and rising edges.

The instruction phase is not affected by DDRM. The instruction is always sent using CLK falling edge.

When receiving data in DDR mode, the QUADSPI assumes that flash memories also send the data using both CLK rising and falling edges. When DDRM = 1, the firmware must clear SSHIFT (bit 4 of QUADSPI_CR). Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Figure 171. Example of a DDR command in quad-SPI mode



Dual-flash mode

When the DFM bit (bit 6 of QUADSPI_CR) is 1, the QUADSPI is in dual-flash mode, where two external quad-SPI flash memories (FLASH 1 and FLASH 2) are used in order to send/receive 8 bits (or 16 bits in DDR mode) every cycle, effectively doubling the throughput as well as the capacity.

Each of the flash memories uses the same CLK and optionally the same NCS signals, but each have separate IO0, IO1, IO2, and IO3 signals.

The dual-flash mode can be used in conjunction with single-, dual-, and quad-SPI modes, as well as with either SDR or DDR mode.

The flash memory size, as specified in FSIZE[4:0] (QUADSPI_DCR[20:16]), must reflect the total flash memory capacity, which is double the size of one individual component.

If address X is even, then the byte which the QUADSPI gives for address X is the byte at the address X/2 of FLASH 1, and the byte which the QUADSPI gives for address X+1 is the byte at the address X/2 of FLASH 2. In other words, bytes at even addresses are all stored in FLASH 1 and bytes at odd addresses are all stored in FLASH 2.

When reading the flash memories status registers in dual-flash mode, twice as many bytes must be read compared to doing the same read in single-flash mode. This means that if each flash memory gives 8 valid bits after the instruction for fetching the status register, then the QUADSPI must be configured with a data length of 2 bytes (16 bits), and the QUADSPI receives one byte from each flash memory. If each flash memory gives a status of 16 bits, then the QUADSPI must be configured to read 4 bytes to get all the status bits of both flash memories in dual-flash mode. The least-significant byte of the result (in the data register) is the least-significant byte of FLASH 1 status register, while the next byte is the least-significant byte of FLASH 2 status register. Then, the third byte of the data register is FLASH 1 second byte, while the forth byte is FLASH 2 second byte (in the case that the flash memories have 16-bit status registers).

An even number of bytes must always be accessed in dual-flash mode. For this reason, bit 0 of the data length bitfield (QUADSPI_DLR[0]) is stuck at 1 when DFM = 1.

In dual-flash mode, the behavior of FLASH 1 interface signals are basically the same as in normal mode. FLASH 2 interface signals have exactly the same waveforms as FLASH 1 during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each flash memory always receives the same instruction and the same address. Then, during the data phase, the BK1_IOx and BK2_IOx buses are both transferring data in parallel, but the data that are sent to (or received from) FLASH 1 are distinct from those of FLASH 2.

26.3.5 QUADSPI indirect mode

When in indirect mode, commands are started by writing to QUADSPI registers, and data are transferred by writing or reading the data register, in the same way as for other communication peripherals.

When FMODE = 00 (QUADSPI_CCR[27:26]), the QUADSPI is in indirect-write mode, where bytes are sent to the flash memory during the data phase. Data are provided by writing to the data register (QUADSPI_DR).

When FMODE = 01, the QUADSPI is in indirect-read mode, where bytes are received from the flash memory during the data phase. Data are recovered by reading QUADSPI_DR.

The number of bytes to be read/written is specified in the data length register (QUADSPI_DLR). If QUADSPI_DLR = 0xFFFF_FFFF (all 1s), then the data length is considered undefined and the QUADSPI simply continues to transfer data until the end of flash memory (as defined by FSIZE) is reached. If no bytes are to be transferred, DMODE (QUADSPI_CCR[25:24]) must be set to 00.

If QUADSPI_DLR = 0xFFFF_FFFF and FSIZE = 0x1F (max value indicating a 4-Gbyte flash memory), then in this special case, transfers continue indefinitely, stopping only after an abort request or after the QUADSPI is disabled. After the last memory address is read (at address 0xFFFF_FFFF), reading continues with address = 0x0000_0000.

When the programmed number of bytes to be transmitted or received is reached, TCF is set and an interrupt is generated if TCIE = 1. In the case of undefined number of data, the TCF is set when the limit of the external SPI memory is reached according to the flash memory size defined in QUADSPI_CR.

Triggering the start of a command

Essentially, a command starts as soon as firmware gives the last information that is necessary for this command. Depending on the QUADSPI configuration, there are three different ways to trigger the start of a command in indirect mode. The commands starts immediately:

- after a write is performed to INSTRUCTION[7:0] (QUADSPI_CCR), if no address is necessary (when ADMODE = 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to ADDRESS[31:0] (QUADSPI_AR), if an address is necessary (when ADMODE ≠ 00) and if no data needs to be provided by the firmware (when FMODE = 01 or DMODE = 00)
- after a write is performed to DATA[31:0] (QUADSPI_DR), if data needs to be provided by the firmware (when FMODE = 00 and DMODE != 00)

Writes to the alternate byte register (QUADSPI_ABR) never trigger the communication start. If alternate bytes are required, they must be programmed before.

As soon as a command is started, BUSY (bit 5 of QUADSPI_SR) is automatically set.

FIFO and data management

In indirect mode, data go through a 32-byte FIFO which is internal to the QUADSPI. FLEVEL[5:0] (QUADSPI_SR[13:8]) indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE = 00), the firmware adds data to the FIFO when it writes QUADSPI_DR. Word writes add 4 bytes to the FIFO, halfword writes add 2 bytes, and byte writes add only 1 byte. If the firmware adds too many bytes to the FIFO (more than is indicated by DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

Byte/halfword accesses to QUADSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES[4:0] is used to define a FIFO threshold. When the threshold is reached, FTF (FIFO threshold flag) is set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there are data in the FIFO after the last byte is read from the flash memory, regardless of the FTHRES setting.

In indirect-write mode, FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when FTF is set. If DMAEN = 1, a DMA transfer is initiated when FTF is set. FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data is transferred by the CPU or DMA).

In indirect-read mode, when the FIFO becomes full, the QUADSPI temporarily stops reading bytes from the flash memory to avoid an overrun. The reading of the flash memory does not restart until 4 bytes become vacant in the FIFO (when FLEVEL ≤ 28). Thus, when FTHRES ≥ 29, the application must take care to read enough bytes to assure that the QUADSPI starts retrieving data from the flash memory again. Otherwise, the FTF flag stays at 0 as long as 28 < FLEVEL < FTHRES.

26.3.6 QUADSPI automatic status-polling mode

In automatic status-polling mode, the QUADSPI periodically starts a command to read a defined number of status bytes (up to 4). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

Accesses to the flash memory begin in the same way as in indirect-read mode: if no address is required (AMODE = 00), accesses begin as soon as the QUADSPI_CCR is written. Otherwise, if an address is required, the first access begins when QUADSPI_AR is written. BUSY goes high at this point and stays high even between the periodic accesses.

The contents of MASK[31:0] (QUADSPI_PSMAR) are used to mask the data from the flash memory in automatic status-polling mode. If MASK[n] = 0, then bit n of the result is masked and not considered. If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] (QUADSPI_PSMAR), then there is a match for bit n.

If the polling match mode bit (PMM, bit 23 of QUADSPI_CR) is 0, then “AND” match mode is activated. This means status match flag (SMF) is set only when there is a match on all of the unmasked bits.

If PMM = 1, then “OR” match mode is activated. This means SMF is set if there is a match on any of the unmasked bits.

An interrupt is called when SMF is set if SMIE = 1.

If the automatic status-polling mode stop (APMS) bit is set, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1, and the periodic accesses continue until there is an abort or the QUADSPI is disabled (EN = 0).

The data register (QUADSPI_DR) contains the latest received status bytes (the FIFO is deactivated). The content of the data register is not affected by the masking used in the matching logic. The FTF status bit is set as soon as a new reading of the status is complete, and FTF is cleared as soon as the data is read.

26.3.7 QUADSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

It is forbidden to access the quad-SPI flash bank area before having properly configured and enabled the QUADSPI peripheral.

No more than 256 Mbytes can addressed even if the flash memory capacity is larger.

If an access is made to an address outside of the range defined by FSIZE but still within the 256-Mbyte range, then a bus error is given. The effect of this error depends on the bus master that attempted the access:

- If it is the Cortex CPU, bus fault exception is generated when enabled (or an HardFault exception when bus fault is disabled).
- If it is a DMA, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

Byte, halfword, and word access types are all supported.

Support for execute in place (XIP) operation is implemented, where the QUADSPI anticipates the next access and load in advance the byte at the following address. If the subsequent access is indeed made at a continuous address, the access is completed faster since the value is already prefetched.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without any access since when the FIFO becomes full with prefetch data.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is a timeout, there is an abort, or the peripheral is disabled.

26.3.8 QUADSPI free-running clock mode

When configured in free-running clock mode, the QUADSPI continuously outputs the clock for test and calibration purposes.

The free-running clock mode is entered as soon as the FRCM bit is set in the QUADSPI_CCR register. It is exited by setting the ABORT bit of the QUADSPI_CR.

When the QUADSPI operates in free-running clock mode:

- the clock is running continuously
- NCS stays high (external device deselected)
- data lines are released (High-Z)
- the BUSY flag of QUADSPI_SR is set

26.3.9 QUADSPI flash memory configuration

The device configuration register (QUADSPI_DCR) can be used to specify the characteristics of the external SPI flash memory.

The FSIZE[4:0] bitfield defines the size of external memory using the following formula:

Number of bytes in flash memory = $2^{[FSIZE+1]}$

FSIZE+1 is effectively the number of address bits required to address the flash memory. The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DFM = 1, FSIZE indicates the total capacity of the two flash memories together.

When the QUADSPI executes two commands, one immediately after the other, it raises NCS high between the two commands for only one CLK cycle by default. If the flash memory requires more time between commands, the CSHT bitfield can be used to specify the minimum number of CLK cycles (up to 8) that NCS must remain high.

The clock mode (CKMODE) bit indicates the CLK signal logic level in between commands (when NCS = 1).

26.3.10 QUADSPI delayed data sampling

By default, the QUADSPI samples the data driven by the flash memory one half of a CLK cycle after the flash memory drives the signal.

In case of external signal delays, it may be beneficial to sample the data later. Using SSHIFT (bit 4 of QUADSPI_CR), the sampling of the data can be shifted by half of a CLK cycle.

Clock shifting is not supported in DDR mode: SSHIFT must be clear when DDRM bit is set.

26.3.11 QUADSPI configuration

The QUADSPI configuration is done in two phases:

1. QUADSPI peripheral configuration
2. QUADSPI flash memory configuration

Once configured and enabled, the QUADSPI can be used in one of its three operating modes: indirect, automatic status-polling, or memory-mapped mode.

QUADSPI configuration

The QUADSPI is configured using QUADSPI_CR. The user must configure the clock prescaler division factor and the sample shifting settings for the incoming data.

The DDR mode can be set through the DDRM bit. When setting the quad-SPI interface in DDR mode, the internal divider of kernel clock must be set with a division ratio of two or more. Once enabled, the address and the alternate bytes are sent on both clock edges, and the data are sent/received on both clock edges. Regardless of the DDRM bit setting, instructions are always sent in SDR mode.

DMA requests are enabled setting the DMAEN bit. In case of interrupt usage, their respective enable bit can be also set during this phase.

The FIFO level for either DMA request generation or interrupt generation is programmed in the FTHRES bits.

If a timeout counter is needed, the TCEN bit can be set and the timeout value programmed in the QUADSPI_LPTR register.

The dual-flash mode can be activated by setting DFM to 1.

QUADSPI flash memory configuration

The parameters related to the targeted external flash memory are configured through the QUADSPI_DCR register.The user must program the flash memory size in the FSIZE bits, the chip-select minimum high time in CSHT bits, and the functional mode (Mode 0 or Mode 3) in the MODE bit.

26.3.12 QUADSPI use

The operating mode is selected using FMODE[1:0] (QUADSPI_CCR[27:26]).

Indirect mode

When FMODE is programmed to 00, the indirect-write mode is selected and data can be sent to the flash memory. With FMODE = 01, the indirect-read mode is selected where data can be read from the flash memory.

When the QUADSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in QUADSPI_DLR.
2. Specify the frame format, mode and instruction code in QUADSPI_CCR.
3. Specify optional alternate byte to be sent right after the address phase in QUADSPI_ABR.
4. Specify the operating mode in QUADSPI_CR. If FMODE = 00 (indirect-write mode) and DMAEN = 1, then QUADSPI_AR must be specified before QUADSPI_CR. Otherwise QUADSPI_DR can be written by the DMA before QUADSPI_AR is updated (if the DMA controller has already been enabled).
5. Specify the targeted address in QUADSPI_AR.
6. Read/write the data from/to the FIFO through QUADSPI_DR.

When writing QUADSPI_CR, the user specifies the following settings:

- enable bit (EN) set to 1
- DMA enable bit (DMAEN) for transferring data to/from RAM
- timeout counter enable bit (TCEN)
- sample shift setting (SSHIFT)
- FIFO threshold level (FTRHES) to indicate when the FTF flag must be set
- interrupt enables
- automatic status-polling mode parameters: match mode and stop mode (valid when FMODE = 11)
- clock prescaler

When writing QUADSPI_CCR, the user specifies the following parameters:

- instruction byte through INSTRUCTION bits
- the way the instruction has to be sent through the IMODE bits (1/2/4 lines)
- the way the address has to be sent through the ADMODE bits (None/1/2/4 lines)
- address size (8/16/24/32-bit) through ADSIZE bits
- the way the alternate bytes have to be sent through the ABMODE (None/1/2/4 lines)
- alternate bytes number (1/2/3/4) through the ABSIZE bits
- presence or not of dummy bytes through the DBMODE bit
- number of dummy bytes through the DCYC bits
- the way data have to be sent/received (none/1/2/4 lines) through DMODE bits

If neither QUADSPI_AR nor QUADSPI_DR need to be updated for a particular command, then the command sequence starts as soon as QUADSPI_CCR is written. This is the case when both ADMODE and DMODE are 00, or if just ADMODE = 00 when in indirect read mode (FMODE = 01).

When an address is required (ADMODE is not 00) and the data register does not need to be written (when FMODE = 01 or DMODE = 00), the command sequence starts as soon as the address is updated with a write to QUADSPI_AR.

In case of data transmission (FMODE = 00 and DMODE! = 00), the communication start is triggered by a write in the FIFO through QUADSPI_DR.

Automatic status-polling mode

This mode is enabled setting the FMODE bitfield (QUADSPI_CCR[27:26]) to 10. In this mode, the programmed frame is sent and the data retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in QUADSPI_DLR, it is ignored and only 4 bytes are read.

The periodicity is specified in the QUADSPI_PISR register.

Once the status data is retrieved, it can internally be processed:

- to set the status match flag and generate an interrupt if enabled
- to stop automatically the periodic retrieving of the status bytes

The received value can be masked with the value stored in QUADSPI_PSMKR and ORed or ANDed with the value stored in QUADSPI_PSMAR.

In case of match, the status match flag is set and an interrupt is generated if enabled, and the QUADSPI can be automatically stopped if the AMPS bit is set.

In any case, the latest retrieved value is available in QUADSPI_DR.

Memory-mapped mode

In memory-mapped mode, the external flash memory is seen as an internal memory but with some latency during accesses. Only read operations are allowed to the external flash memory in this mode.

The memory-mapped mode is entered by setting FMODE to 11 in QUADSPI_CCR.

The programmed instruction and frame is sent when a master is accessing the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate linear reads. Any access to QUADSPI_DR in this mode returns zero.

QUADSPI_DLR has no meaning in memory-mapped mode.

26.3.13 Sending the instruction only once

Some flash memories (for example: Winbound) provide a mode where an instruction must be sent only with the first command sequence, while subsequent commands start directly with the address. One can take advantage of such a feature using the SIOO bit (QUADSPI_CCR[28]).

SIOO is valid for all functional modes (indirect, automatic status-polling, and memory-mapped). If the SIOO bit is set, the instruction is sent only for the first command following a write to QUADSPI_CCR. Subsequent command sequences skip the instruction phase, until there is a write to QUADSPI_CCR.

SIOO has no effect when IMODE = 00 (no instruction).

26.3.14 QUADSPI error management

An error can be generated in the following case:

- In indirect mode or automatic status-polling mode when a wrong address is programmed in QUADSPI_AR (according to the flash memory size defined by FSIZE[4:0] in the QUADSPI_DCR), TEF is set and an interrupt is generated if enabled.

  Also in indirect mode, if the address plus the data length exceeds the flash memory size, TEF is set as soon as the access is triggered.
- In memory-mapped mode, when an out-of-range access is done by a master or when the QUADSPI is disabled, a bus error is generated as a response to the faulty bus master request.

  When a master is accessing the memory mapped space while the memory-mapped mode is disabled, a bus error is generated as a response to the faulty bus master request.

26.3.15 QUADSPI busy bit and abort functionality

Once the QUADSPI starts an operation with the flash memory, the BUSY bit is automatically set in QUADSPI_SR.

In indirect mode, BUSY is reset once the QUADSPI has completed the requested command sequence, and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1, or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on a timeout event or on an abort.

Any operation can be aborted by setting the ABORT bit in QUADSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

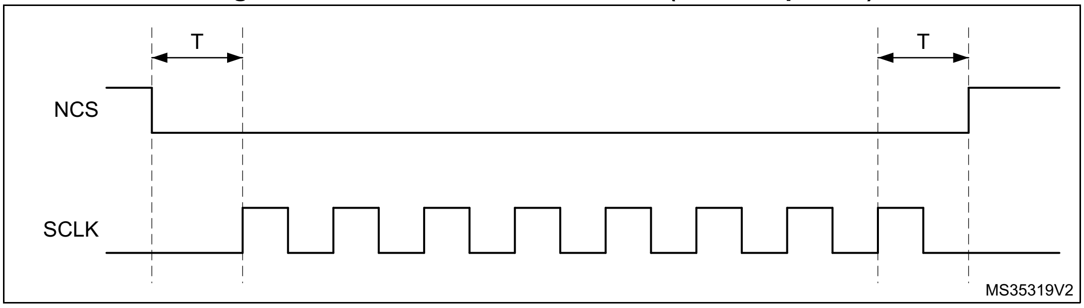
Note: Some flash memories may misbehave if a write operation to a status registers is aborted.

26.3.16 NCS behavior

By default, NCS is high, deselecting the external flash memory. NCS falls before an operation begins and rises as soon as it finishes.

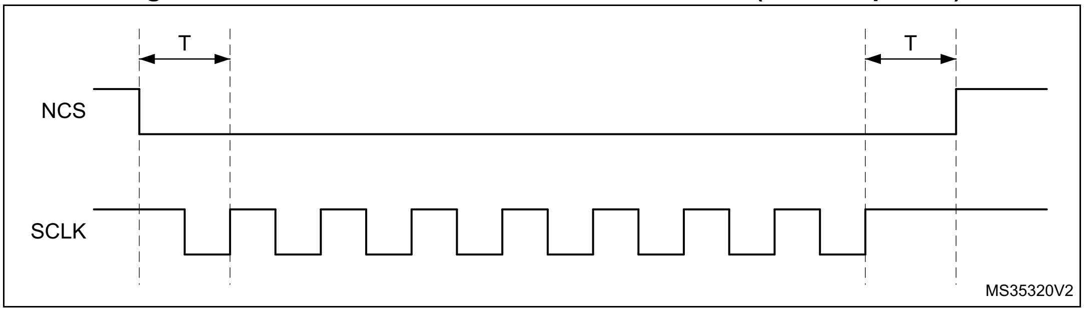
When CKMODE = 0 (“mode0”, where CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

Figure 172. NCS when CKMODE = 0 (T = CLK period)



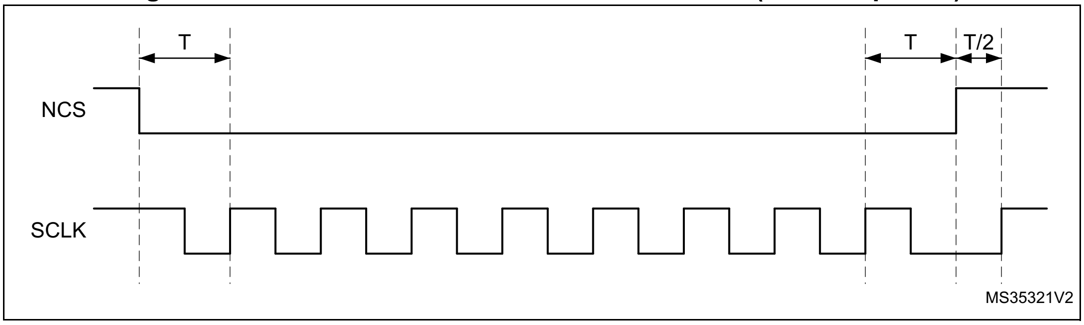
When CKMODE=1 (“mode3”, where CLK goes high when no operation is in progress) and DDRM=0 (SDR mode), NCS still falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge, as shown in the figure below.

Figure 173. NCS when CKMODE = 1 in SDR mode (T = CLK period)



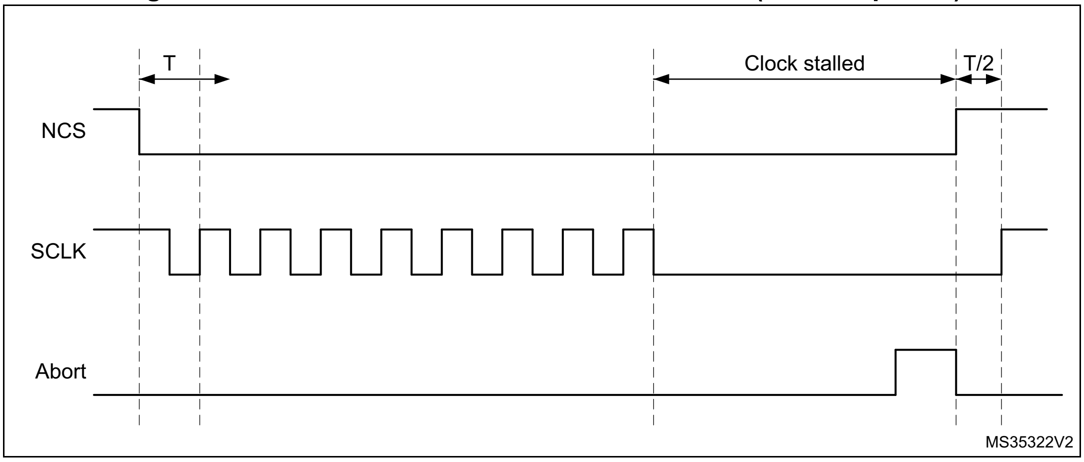
When CKMODE = 1 (“mode3”) and DDRM = 1 (DDR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge, as shown in the figure below. Because DDR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 174. NCS when CKMODE = 1 in DDR mode (T = CLK period)



When the FIFO stays full in a read operation or if the FIFO stays empty in a write operation, the operation stalls and CLK stays low until firmware services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later, as shown in *Figure 175*.

Figure 175. NCS when CKMODE = 1 with an abort (T = CLK period)



When not in dual-flash mode (DFM = 0) and FSEL = 0 (default value), only FLASH 1 is accessed. Thus BK2_NCS stays high, if FSEL = 1, only FLASH 2 is accessed and BK1_NCS stays high. In dual-flash mode, BK2_NCS behaves exactly the same as BK1_NCS. Thus, if there is a FLASH 2 and if the application is dual-flash mode only, then BK1_NCS signal can be used for FLASH 2 as well, and the pin devoted to BK2_NCS can be used for other functions.

26.4 QUADSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 178. QUADSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

26.5 QUADSPI registers

26.5.1 QUADSPI control register (QUADSPI_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESCALER[7:0]								PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	FTHRES[4:0]					FSEL	DFM	Res.	SSHIFT	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:24 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating CLK based on the quadspi_ker_ck clock (value+1).

0: $F_{CLK} = F_{quadspi\_ker\_ck}$, quadspi_ker_ck clock used directly as QUADSPI CLK (prescaler bypassed)

1: $F_{CLK} = F_{quadspi\_ker\_ck}/2$

2: $F_{CLK} = F_{quadspi\_ker\_ck}/3$

...

255: $F_{CLK} = F_{quadspi\_ker\_ck}/256$

For odd clock division factors, CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.

When setting quad-SPI interface in DDR mode, the prescaler must be set with a division ratio of two or more.

Note: This bitfield can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode

This bit indicates which method must be used for determining a “match” during automatic status-polling mode.

0: AND match mode. SMF is set if all the unmasked bits received from the flash memory match the corresponding bits in the match register.

1: OR match mode. SMF is set if any one of the unmasked bits received from the flash memory matches its corresponding bit in the match register.

Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop

This bit determines if automatic status-polling is stopped after a match.

0: Automatic status-polling mode is stopped only by abort or by disabling the QUADSPI.

1: Automatic status-polling mode stops as soon as there is a match.

Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable

This bit enables the timeout interrupt.

0: Interrupt disabled

1: Interrupt enabled

Bit 19 **SMIE**: Status match interrupt enable
This bit enables the status match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level
This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag (bit FTF in register QUADSPI_SR) to be set.
0: In indirect-write mode (FMODE = 00), FTF is set if there are one or more free bytes location left in the FIF,O or indirect-read mode (FMODE = 01), FTF is set if there are one or more valid bytes that can be read from the FIFO.
1: In indirect-write mode (FMODE = 00), FTF is set if there are two or more free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are two or more valid bytes that can be read from the FIFO
...
31: In indirect-write mode (FMODE = 00), FTF is set if there are 32 free bytes location left in the FIFO, or indirect-read mode (FMODE = 01), FTF is set if there are 32 valid bytes that can be read from the FIFO.
If DMAEN = 1, then the DMA controller for the corresponding channel must be disabled before changing the FTHRES value.

Bit 7 **FSEL**: Flash memory selection
This bit selects the flash memory to be addressed in single-flash mode (when DFM = 0).
0: FLASH 1 selected
1: FLASH 2 selected
Note: This bit can be modified only when BUSY = 0. This bit is ignored when DFM = 1.

Bit 6 **DFM**: Dual-flash mode
This bit activates dual-flash mode, where two external flash memories are used simultaneously to double throughput and capacity.
0: Dual-flash mode disabled
1: Dual-flash mode enabled
Note: This bit can be modified only when BUSY = 0.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **SSHIFT**: Sample shift

By default, the QUADSPI samples data 1/2 of a CLK cycle after the data is driven by the flash memory. This bit allows the data to be sampled later in order to account for external signal delays.

0: No shift

1: 1/2 cycle shift

The firmware must assure that SSHIFT = 0 when in DDR mode (when DDRM = 1).

Note: This bitfield can be modified only when BUSY = 0.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when memory-mapped mode (FMODE = 11) is selected. Activating this bit causes the NCS to be released (and thus reduces consumption) if there has not been an access after a certain amount of time, where this time is defined by TIMEOUT[15:0] (QUADSPI_LPTR). This bit enables the timeout counter.

By default, the QUADSPI never stops its prefetch operation, keeping the previous read operation active with NCS maintained low, even if no access to the flash memory occurs for a long time. Since flash memories tend to consume more when NCS is held low, the application may want to activate the timeout counter (TCEN = 1, bit 3 of QUADSPI_CR) so that NCS is released after a period of TIMEOUT[15:0] (QUADSPI_LPTR) cycles have elapsed without an access since when the FIFO becomes full with prefetch data.

0: Timeout counter is disabled, and thus the NCS remains active indefinitely after an access in memory-mapped mode.

1: Timeout counter is enabled, and thus the NCS is released in memory-mapped mode after TIMEOUT[15:0] cycles of flash memory inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via the QUADSPI_DR register. DMA transfers are initiated when the FIFO threshold flag, FTF, is set.

0: DMA is disabled for indirect mode.

1: DMA is enabled for indirect mode.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is complete. This bit stops the current transfer.

In automatic status-polling or memory-mapped mode, this bit also reset APM or DM bit.

0: No abort requested

1: Abort requested

Bit 0 **EN**: QUADSPI enable

0: QUADSPI disabled

1: QUADSPI enabled

26.5.2 QUADSPI device configuration register (QUADSPI_DCR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSIZE[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CSHT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKMODE
					rw	rw	rw								rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **FSIZE[4:0]**: Flash memory size

This bitfield defines the size of external memory using the following formula:

Number of bytes in flash memory = $2^{[FSIZE+1]}$

FSIZE+1 is effectively the number of address bits required to address the flash memory. The flash memory capacity can be up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DFM = 1, FSIZE indicates the total capacity of the two flash memories together.

Note: This bitfield can be modified only when BUSY = 0.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **CSHT[2:0]**: Chip select high time

CSHT+1 defines the minimum number of CLK cycles which the chip select (NCS) must remain high between commands issued to the flash memory.

0: NCS stays high for at least 1 cycle between flash memory commands

1: NCS stays high for at least 2 cycles between flash memory commands

...

7: NCS stays high for at least 8 cycles between flash memory commands

Note: This bitfield can be modified only when BUSY = 0.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **CKMODE**: Mode 0/mode 3

This bit indicates the level that CLK takes between commands (when NCS = 1).

0: CLK must stay low while NCS is high (chip select released). This is referred to as mode 0.

1: CLK must stay high while NCS is high (chip select released). This is referred to as mode 3.

Note: This bitfield can be modified only when BUSY = 0.

26.5.3 QUADSPI status register (QUADSPI_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
		r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes which are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full. In memory-mapped mode and in automatic status-polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is on going. This bit clears automatically when the operation with the flash memory is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (QUADSPI_PSMAR). It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold is reached, or if there is any data left in the FIFO after reads from the flash memory are complete. It is cleared automatically as soon as threshold condition is no longer true.

In automatic status-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data is transferred or in any mode when the transfer is aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode. It is cleared by writing 1 to CTEF.

26.5.4 QUADSPI flag clear register (QUADSPI_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
											w	w		w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CTOF**: Clear timeout flag

Writing 1 clears the TOF flag in QUADSPI_SR.

Bit 3 **CSMF**: Clear status match flag

Writing 1 clears the SMF flag in QUADSPI_SR.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag

Writing 1 clears the TCF flag in QUADSPI_SR.

Bit 0 **CTEF**: Clear transfer error flag

Writing 1 clears the TEF flag in QUADSPI_SR.

26.5.5 QUADSPI data length register (QUADSPI_DLR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DL[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value no greater than 3 (indicating 4 bytes) must be used for automatic status-polling mode.

All 1s in indirect mode means undefined length, where the QUADSPI continues until the end of memory, as defined by FSIZE.

0x0000_0000: 1 byte is to be transferred

0x0000_0001: 2 bytes are to be transferred

0x0000_0002: 3 bytes are to be transferred

0x0000_0003: 4 bytes are to be transferred

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred

0xFFFF_FFFF: undefined length -- all bytes until the end of flash memory (as defined by FSIZE) are to be transferred. Continue reading indefinitely if FSIZE = 0x1F.

DL[0] is stuck at 1 in dual-flash mode (DFM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect when in memory-mapped mode (FMODE = 10).

Note: This bitfield can be written only when BUSY = 0.

26.5.6 QUADSPI communication configuration register (QUADSPI_CCR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDRM	DHHC	FRCM	SIOO	FMODE[1:0]		DMODE[1:0]		Res.	DCYC[4:0]					ABSIZE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABMODE[1:0]		ADSIZE[1:0]		ADMODE[1:0]		IMODE[1:0]		INSTRUCTION[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DDRM**: Double data rate mode

This bit sets the DDR mode for the address, alternate byte and data phase:

0: DDR mode disabled

1: DDR mode enabled

Note: This bit can be written only when BUSY = 0.

Bit 30 **DHHC**: DDR hold

This bit delays the data output by 1/4 of the QUADSPI output clock cycle in DDR mode:

0: Data output delayed using analog delay

1: Data output delayed by 1/4 of a QUADSPI output clock cycle

This feature is only active in DDR mode.

Note: This bit can be written only when BUSY = 0.

PRESCALER > 0 is mandatory when DHHC = 1.

Bit 29 **FRCM:** Free-running clock mode

When this bit is set, the QUADSPI enters free-running clock mode regardless of FMODE bits.

0: Normal mode

1: Free-running clock mode

Note: This bit can be written only when BUSY = 0.

Bit 28 **SIOO**: Send instruction only once mode

This bit has no effect when IMODE = 00.See *Section 26.3.13* for more details.

0: Instruction sent on every transaction

1: Instruction sent only for the first command

Note: This bit can be written only when BUSY = 0.

Bits 27:26 **FMODE[1:0]**: Functional mode

This bitfield defines the QUADSPI functional mode of operation.

00: Indirect-write mode

01: Indirect-read mode

10: Automatic status-polling mode

11: Memory-mapped mode

If DMAEN = 1 already, the DMA controller for the corresponding channel must be disabled before changing the FMODE value.

Note: This bitfield can be written only when BUSY = 0.

Bits 25:24 **DMODE[1:0]**: Data mode

This bitfield defines the data phase mode of operation:

00: No data

01: Data on a single line

10: Data on two lines

11: Data on four lines

This bitfield also determines the dummy phase mode of operation.

Note: This bitfield can be written only when BUSY = 0.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **DCYC[4:0]**: Number of dummy cycles

This bitfield defines the duration of the dummy phase. In both SDR and DDR modes, it specifies a number of CLK cycles (0-31).

Note: This bitfield can be written only when BUSY = 0.

Bits 17:16 **ABSIZE[1:0]**: Alternate-byte size

This bit defines the size of alternate bytes.

00: 8-bit alternate byte

01: 16-bit alternate bytes

10: 24-bit alternate bytes

11: 32-bit alternate bytes

Note: This bitfield can be written only when BUSY = 0.

Bits 15:14 **ABMODE[1:0]**: Alternate byte mode

This bitfield defines the alternate-byte phase mode of operation.

00: No alternate bytes

01: Alternate bytes on a single line

10: Alternate bytes on two lines

11: Alternate bytes on four lines

Note: This bitfield can be written only when BUSY = 0.

Bits 13:12 **ADSIZE[1:0]**: Address size

This bit defines address size:

00: 8-bit address

01: 16-bit address

10: 24-bit address

11: 32-bit address

Note: This bitfield can be written only when BUSY = 0.

Bits 11:10 **ADMODE[1:0]**: Address mode
This bitfield defines the address phase mode of operation.
00: No address
01: Address on a single line
10: Address on two lines
11: Address on four lines
Note: This bitfield can be written only when BUSY = 0.

Bits 9:8 **IMODE[1:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
00: No instruction
01: Instruction on a single line
10: Instruction on two lines
11: Instruction on four lines
Note: This bitfield can be written only when BUSY = 0.

Bits 7:0 **INSTRUCTION[7:0]**: Instruction
Instruction to be sent to the external SPI device.
Note: This bitfield can be written only when BUSY = 0.

26.5.7 QUADSPI address register (QUADSPI_AR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address
This bitfield contains the address to be sent to the external flash memory.
Writes to this bitfield are ignored when BUSY = 1 or when FMODE = 11 (memory-mapped mode).
In dual flash mode, ADDRESS[0] is automatically stuck to 0 as the address must always be even

26.5.8 QUADSPI alternate-byte register (QUADSPI_ABR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALTERNATE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTERNATE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be send to the external SPI device right after the address.

Note: This bitfield can be written only when BUSY = 0.

26.5.9 QUADSPI data register (QUADSPI_DR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device.

In indirect write mode, data written to this register is stored on the FIFO before it is sent to the flash memory during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect read mode, reading this register gives (via the FIFO) the data which was received from the flash memory. If the FIFO does not have as many bytes as requested by the read operation and if BUSY=1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the flash memory (without masking).

Word, halfword, and byte accesses to this register are supported. In indirect write mode, a byte write adds 1 byte to the FIFO, a halfword write 2, and a word write 4. Similarly, in indirect read mode, a byte read removes 1 byte from the FIFO, a halfword read 2, and a word read 4. Accesses in indirect mode must be aligned to the bottom of this register: a byte read must read DATA[7:0] and a halfword read must read DATA[15:0].

26.5.10 QUADSPI polling status mask register (QUADSPI_PSMKR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode.

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic

Note: This bitfield can be written only when BUSY = 0.

26.5.11 QUADSPI polling status match register (QUADSPI_PSMAR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MATCH[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MATCH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match.

Note: This bitfield can be written only when BUSY = 0.

26.5.12 QUADSPI polling interval register (QUADSPI_PIR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTERVAL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval

Number of CLK cycles between two read during automatic status-polling phases.

Note: This bitfield can be written only when BUSY = 0.

26.5.13 QUADSPI low-power timeout register (QUADSPI_LPTR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TIMEOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the QUADSPI prefetches the subsequent bytes and holds these bytes in the FIFO. This bitfield indicates how many CLK cycles the QUADSPI waits after the FIFO becomes full until it raises NCS, putting the flash memory in a lower-consumption state.

Note: This bitfield can be written only when BUSY = 0.

26.5.14 QUADSPI hardware configuration register (QUADSPI_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 B058

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IDLENGTH[3:0]				PRESCVAL[3:0]				FIFOPTR[3:0]				FIFOSIZE[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IDLENGTH[3:0]**: ID length

This bitfield returns the length of the AXI IDs.

Bits 11:8 **PRESCVAL[3:0]**: Prescaler reset value

This bitfield returns the reset value of the prescaler.

Bits 7:4 **FIFOPTR[3:0]**: FIFO pointer size

This bitfield returns the size in bit of the FIFO pointer.

Bits 3:0 **FIFOSIZE[3:0]**: FIFO size

This bitfield returns the size of FIFO in words.

26.5.15 QUADSPI version register (QUADSPI_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: QUADSPI major revision

Bits 3:0 **MINREV[3:0]**: QUADSPI minor revision

26.5.16 QUADSPI identification register (QUADSPI_IPIDR)

Address offset: 0x3F8

Reset value: 0x0014 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: QUADSPI identification code

26.5.17 QUADSPI size identification register (QUADSPI_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: QUADSPI size and identification code

26.5.18 QUADSPI register map

Table 179. QUADSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	QUADSPI_CR	PRESCALER[7:0]								PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					FSEL	DFM	Res.	SSHIFT	TCEN	DMAEN	ABORT	EN
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0				0	0	0	0	0	0	0		0	0	0	0	0
0x004	QUADSPI_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSIZE[4:0]					Res.	Res.	Res.	Res.	Res.	CSHT			Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKMODE
	Reset value												0	0	0	0	0						0	0	0								0
0x008	QUADSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x00C	QUADSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x010	QUADSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	QUADSPI_CCR	DDRM	DHHC	FRCM	SIOO	FMODE [1:0]		DMODE [1:0]		Res.	DCYC[4:0]					ABSIZE [1:0]		ABMODE [1:0]		ADSIZE [1:0]		ADMODE [1:0]		IMODE [1:0]		INSTRUCTION[7:0]							
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	QUADSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	QUADSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	QUADSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	QUADSPI_ PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	QUADSPI_ PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	QUADSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	QUADSPI_ LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034-0x3EC	Reserved	Reserved																															
0x3F0	QUADSPI_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLENGTH [3:0]				PRESCVAL [3:0]				FIFOPTR [3:0]				FIFOSIZE [3:0]			
	Reset value	0x0000 B058																															
0x3F4	QUADSPI _VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value	0x0000 0041																															
0x3F8	QUADSPI_IDR	ID[31:0]																															
	Reset value	0x0014 0031																															
0x3FC	QUADSPI_SIDR	SID[31:0]																															
	Reset value	0xA3C5 DD01																															

Refer to *Section 2.5* for the register boundary addresses.

27 Delay block (DLYB)

27.1 Introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or Quad-SPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

27.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

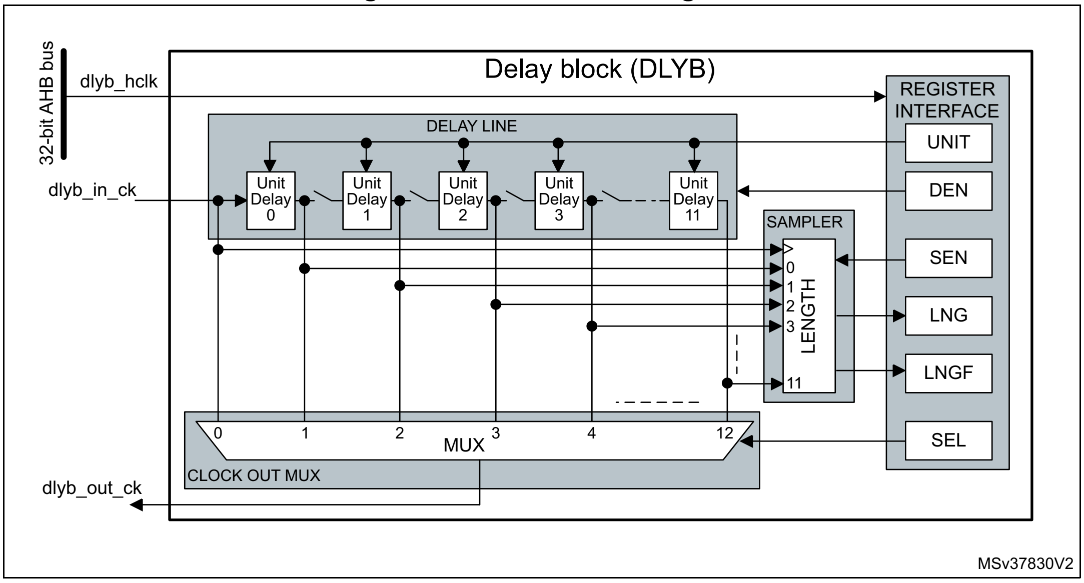
27.3 DLYB functional description

27.3.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 176. DLYB block diagram



27.3.2 DLYB pins and internal signals

Table 180 lists the DLYB internal signals.

Table 180. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

Table 181. DLYB interconnection

Signal name	Source/destination			
	SDMMC1	SDMMC2	SDMMC3	QUADSPI
dlyb_in_ck	sdmmc_io_in_ck			qspi_clk_o from IO pad
dlyb_out_ck	sdmmc_fb_ck			qspi_clk_fb_i

27.3.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 182. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

27.3.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

27.3.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

SDMMC use case:

The delay block is used in conjunction with SDMMC interface variable delay. For correct sampling point tuning the delay value must cover a whole SDMMC_CK clock period. After having tuned the delay line length the individual delays are used in the sampling point tuning to find the optimal sampling point.

27.4 DLYB registers

All registers can be accessed in word, half-word and byte access.

27.4.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

27.4.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LNGF	Res.	Res.	Res.	LNG[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock.

The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.

These bits can only be written when SEN = 1.

Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.

These bits can only be written when SEN = 1.

Output clock phase = input clock + SEL[3:0] x unit delay

27.4.3 DLYB register map

Table 183. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.5* for the register boundary addresses.

28 Analog-to-digital converters (ADC)

28.1 Introduction

This section describes the ADC implementation:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 16-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows to improve analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

28.2 ADC main features

- High-performance features
  - Up to 2x ADCs which can operate in dual mode
  - 16, 14, 12, 10 or 8-bit configurable resolution
  - ADC conversion time is independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Can manage single-ended or differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Self-calibration (both offset and linearity)
  - Channel-wise programmable sampling time
  - Up to four injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by GP-DMA for regular channel conversions with FIFO
  - Data can be routed to DFSDM for post processing
  - 4 dedicated data registers for the injected channels
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024x
  - Programmable data right and left shift
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- Each ADC features an external analog input channel
  - Up to 6 fast channels from dedicated GPIO pads
  - Up to 14 slow channels from dedicated GPIO pads
- In addition, there are 6 internal dedicated channels
  - Internal reference voltage (V_{REFINT})
  - Internal temperature sensor (V_{SENSE})
  - V_{BAT} monitoring channel ($V_{BAT}/4$)
  - Connection to DAC internal channels
  - V_{DDCORE} monitoring channel
- Start-of-conversion can be initiated:
  - by software for both regular and injected conversions
  - by hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes

- Each ADC can convert a single channel or can scan a sequence of channels
- Single mode converts selected inputs once per trigger
- Continuous mode converts selected inputs continuously
- Discontinuous mode

- Dual ADC mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

Figure 177 shows the block diagram of one ADC.

28.3 ADC implementation

Table 184. ADC features

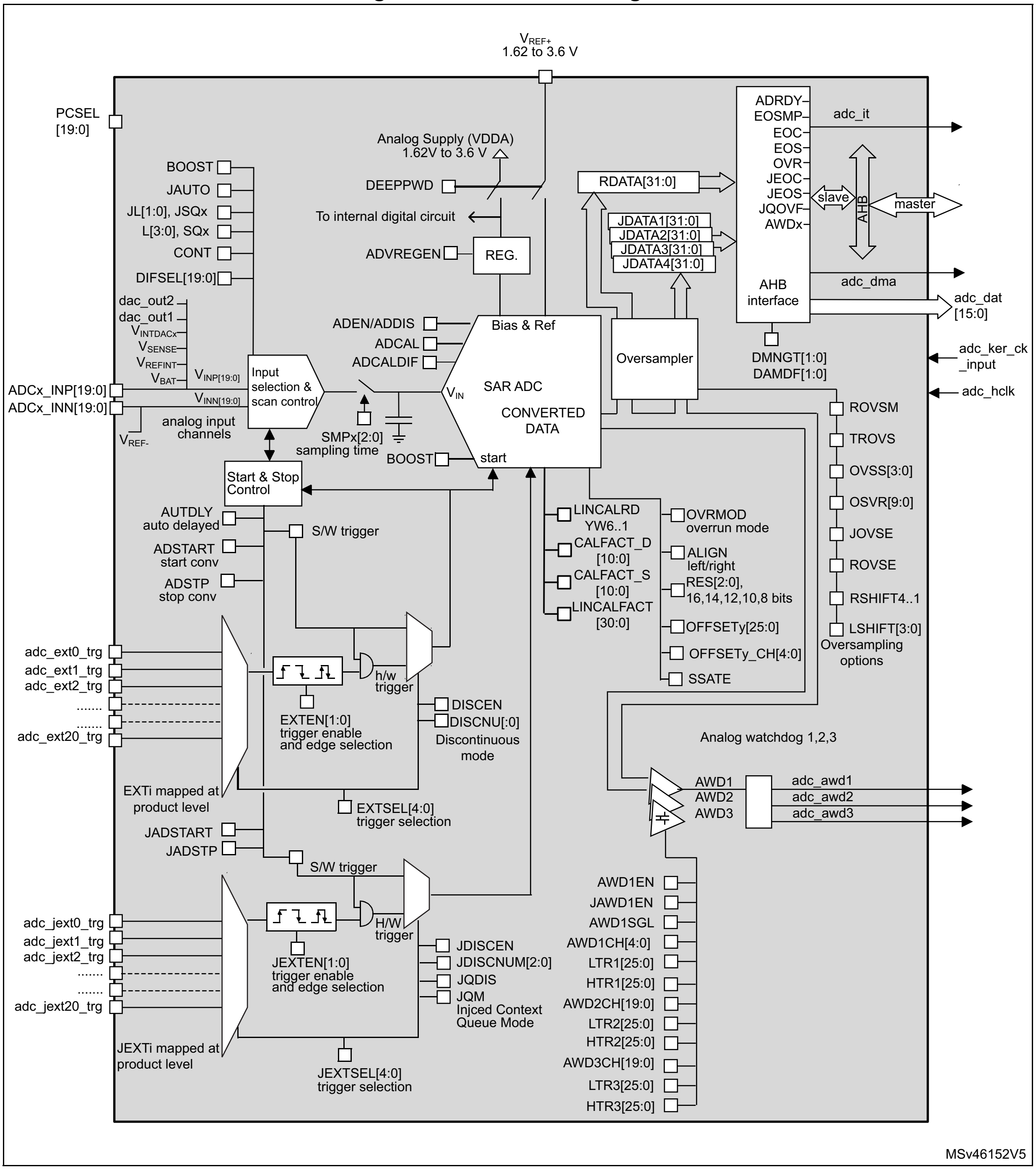
ADC modes/features	ADC1	ADC2
Dual mode	X (coupled together)	
DFSDM interface	X	X
Extended sample time option (SMPPLUS control)	-	-
LDO voltage regulator status	X	X

28.4 ADC functional description

28.4.1 ADC block diagram

Figure 177 shows the ADC block diagram and *Table 185* gives the ADC pin description.

Figure 177. ADC block diagram



28.4.2 ADC pins and internal signals

Table 185. ADC input/output pins

Name	Signal type	Description
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VDDA	Input, analog supply	Analog power supply equal V_{DDA}
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC.
VSSA	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_INPy	External analog inputs	Up to 20 analog input channels (x = ADC number= 1 to 2): – ADCx_INP[0:5] fast channels – ADCx_INP[6:19] slow channels
ADCx_INNy		Up to 20 analog input channels (x = ADC number= 1 to 2): – ADCx_INN[0:5] fast channels – ADCx_INN[6:19] slow channels

Table 186. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}[y]$	Analog inputs	Positive input analog channels for each ADC, connected either to ADCx_INP*i* external channels or to internal channels.
$V_{INN}[y]$	Analog inputs	Negative input analog channels for each ADC, connected either to V_{REF-} or to ADCx_INN*i* external channels
adc_ext_trgy	Inputs	Up to 21 external trigger inputs for the regular conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgy	Inputs	Up to 21 external trigger inputs for the injected conversions (can be connected to on-chip timers). These inputs are shared between the ADC master and the ADC slave.
adc_awd1 adc_awd2 adc_awd3	Outputs	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
adc_it	Output	ADC interrupt
adc_hclk	Input	AHB clock
adc_ker_ck_input	Input	ADC kernel clock input from RCC
adc_dma	Output	ADC DMA requests
adc_dat[15:0]	Outputs	ADC data outputs

Table 187. ADC interconnection

Signal name	Source/destination
ADC2 $V_{INP}[12]$	V_{SENSE} (output voltage from internal temperature sensor)
ADC2 $V_{INP}[13]$	V_{REFINT} (output voltage from internal reference voltage)
ADC2 $V_{INP}[15]$	$V_{BAT}/4$ (external battery voltage supply voltage)
ADC2 $V_{INP}[14]$	V_{DDCORE} (core logic supply voltage)
ADC2 $V_{INP}[16]$	dac_out1
ADC2 $V_{INP}[17]$	dac_out2
adc_dat[15:0]	dfsdm_dat_adc[15:0]

28.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The input clock is the same for the all ADCs and can be selected between two different clock sources (see *Figure 178: ADC clock scheme*):

1. The ADC clock can be a specific clock source, named adc_ker_ck which is independent and asynchronous with the AHB clock.

   It can be configured in the RCC (refer to RCC Section for more information on how to generate the ADC clock (adc_ker_ck) dedicated clock).

   To select this scheme, CKMODE[1:0] bits of the ADCx_CCR register must be reset.
2. The ADC clock can be derived from the AHB clock of the ADC bus interface, divided by a programmable factor (1, 2 or 4). In this mode, a programmable divider factor can be selected (/1, 2 or 4 according to bits CKMODE[1:0]).

   To select this scheme, CKMODE[1:0] bits of the ADCx_CCR register must be different from “00”.

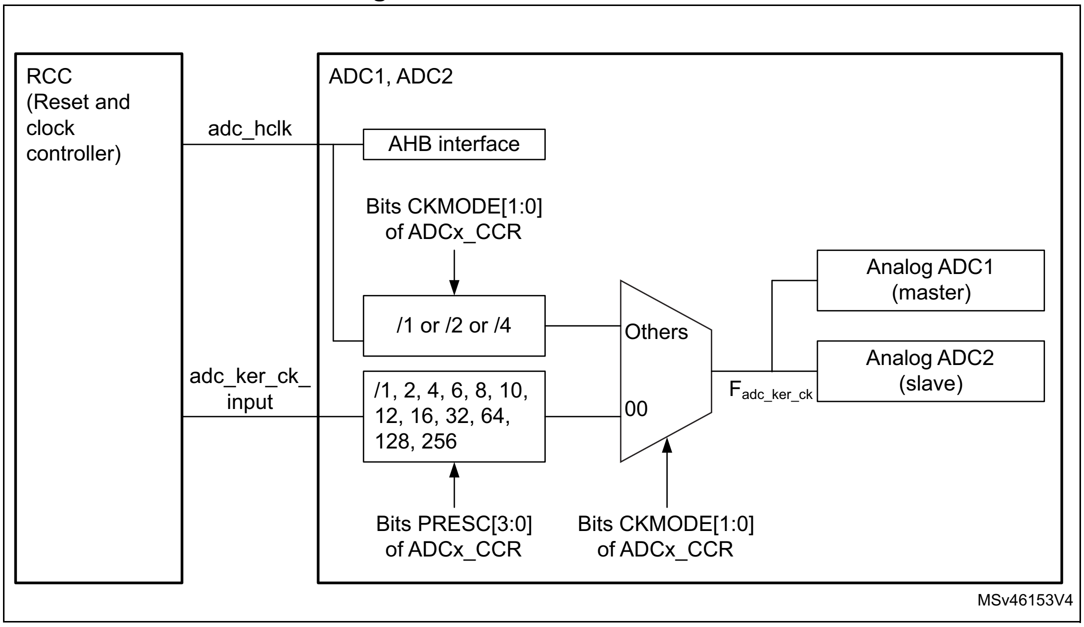
Note: For option 2), a prescaling factor of 1 (CKMODE[1:0]=01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0XXX in RCC_CFGR register, corresponding to a clock duty cycle of 50 %).

Option 1) has the advantage of reaching the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256; using the prescaler configured with bits PRESC[3:0] in the ADCx_CCR register.

Option 2) has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the product datasheet.

Figure 178. ADC clock scheme



1. Refer to the RCC section to see how adc_hclk and adc_ker_ck can be generated.

Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc\_hclk} \geq F_{adc\_ker\_ck} / 4$ if the resolution of all channels are 16-bit, 14-bit,12-bit or 10-bit
- $F_{adc\_hclk} \geq F_{adc\_ker\_ck} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)

Constraints between ADC clocks

When several ADC interfaces are used simultaneously, it is mandatory to use the same clock source from the RCC block without prescaler ratio, for all ADC interfaces.

BOOST control

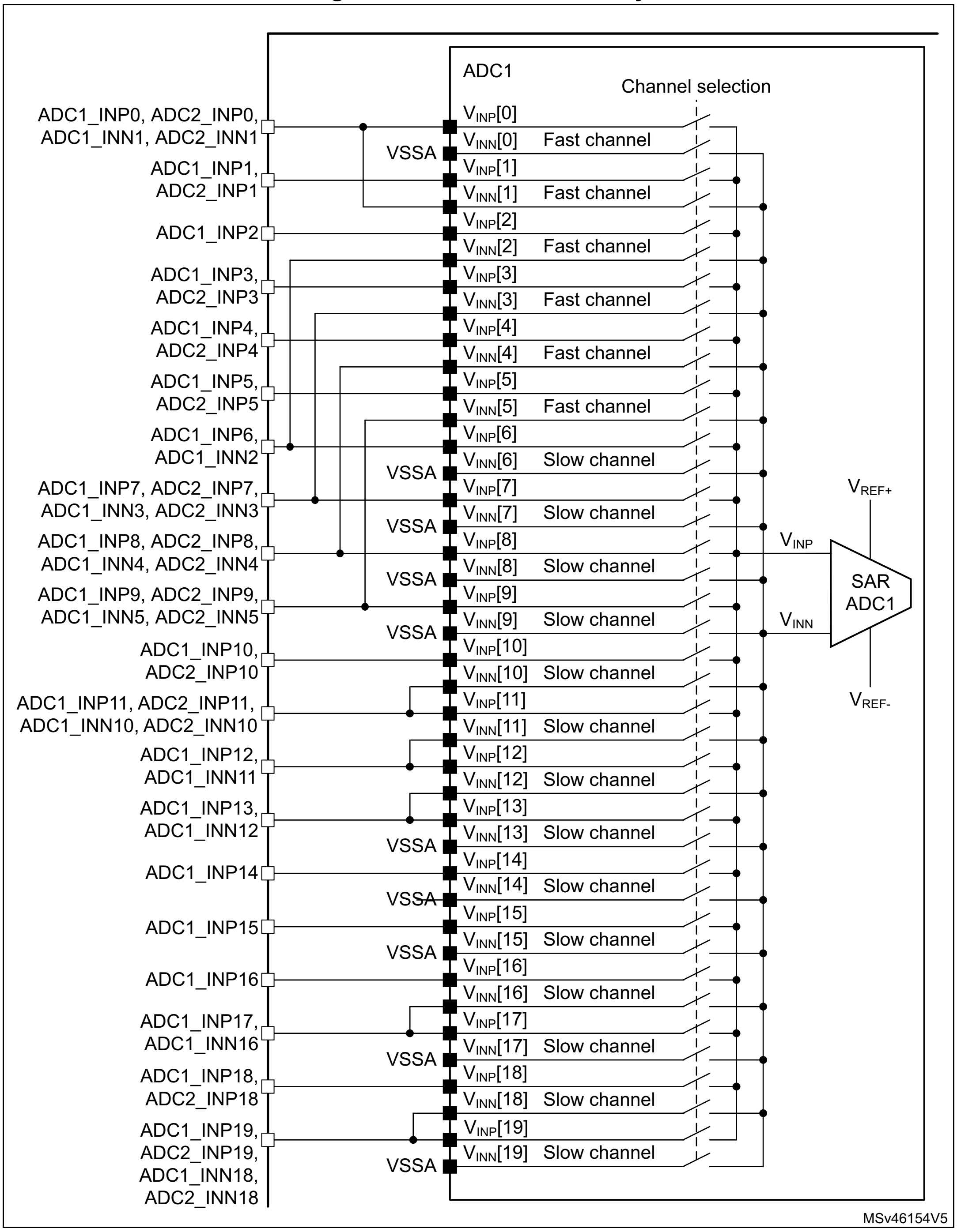
The ADC boost mode can be controlled through the BOOST bit in the ADC_CR register.

This bit must be set according to the ADC clock setting. Refer to the ADC_CR register description.

28.4.4 ADC1/2 connectivity

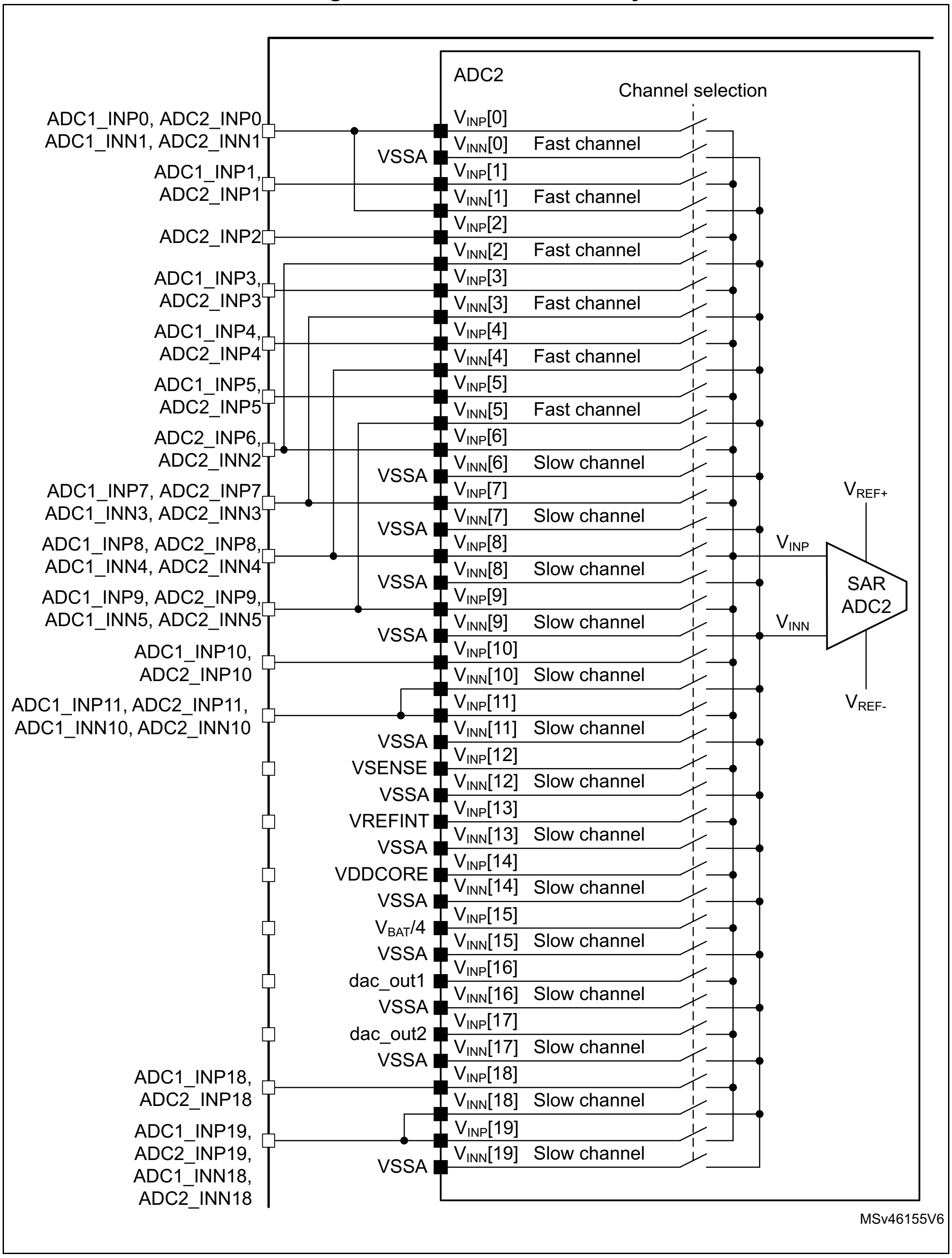
ADC1 and ADC2 are tightly coupled and share some external channels as described in the following figures.

Figure 179. ADC1 connectivity



1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.

Figure 180. ADC2 connectivity



1. ADCx_INNy signal can only be used when the corresponding ADC input channel is configured as differential mode.

28.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

28.4.6 ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit deep-power-down mode by clearing bit DEEPPWD=0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN=1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG\_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software.

The LDO status can be verified by checking the LDORDY bit in ADC_ISR register (refer to *Section 28.3: ADC implementation* for the availability of the LDO regulator status).

For the startup time of the ADC voltage regulator, refer to device datasheet for $T_{ADCVREG\_STUP}$ parameter.

After ADC operations are complete, the ADC can be disabled (ADEN=0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN=0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC deep-power-down mode by setting bit DEEPPWD=1 into ADC_CR register. This is particularly interesting before entering Stop mode.

Note: Writing DEEPPWD=1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

Note: When the internal voltage regulator is disabled (ADVREGEN=0), the internal analog calibration is kept.

In ADC deep-power-down mode (DEEPPWD=1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or apply again the calibration factor which was previously saved (refer to *Section 28.4.8: Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT)*).

28.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by writing into bits DIFSEL[19:0] in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN=0).

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

The output data for the differential mode is an unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x0000 (16-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFFF.

$$\text{Converted value} = \frac{\text{ADC\_Full\_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When ADC is configured as differential mode, both input should be biased at V_{REF+} / 2 voltage.

The input signal are supposed to be differential (common mode voltage should be fixed).

For a complete description of how the input channels are connected for each ADC, refer to *Section 28.4.4: ADC1/2 connectivity*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage is connected to $V_{INN[i]}$. As a consequence, channel “*i+n*”, which is connected to $V_{INN[i]}$, should not be converted at same time by different ADCs. Some channels are shared between ADC1/ADC2: this can make the channel on the other ADC unusable.

28.4.8 Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 11-bits of offset or 160-bits of linearity and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

The calibration is preliminary to any ADC operation. It removes the systematic errors which may vary from chip to chip and allows to compensate offset and linearity deviation.

The calibration factor for the offset to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration which is applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration which is applied for differential input conversions.

The linearity correction must be done once only, regardless of single / differential configuration.

- Write ADCALLIN = 1 before launching a calibration which runs the linearity calibration same time as the offset calibration.
- Write ADCALLIN = 0 before launching a calibration which does not run the linearity calibration but only the offset calibration.

The calibration is then initiated by software by setting bit ADCAL=1. It can be initiated only when the ADC is disabled (when ADEN=0). ADCAL bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[10:0] or CALFACT_D[10:0] of ADC_CALFACT register (depending on single-ended or differential input calibration). The 160-bit linearity calibration factor can be accessed using the ADC_CALFACT2 register with ADEN set to 1.

The internal analog calibration is kept if the ADC is disabled (ADEN=0). However, if the ADC is disabled for extended periods, it is recommended that a new offset calibration cycle is run before enabling again the ADC.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in STANDBY or V_{BAT} mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT and ADC_CALFACT2 register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

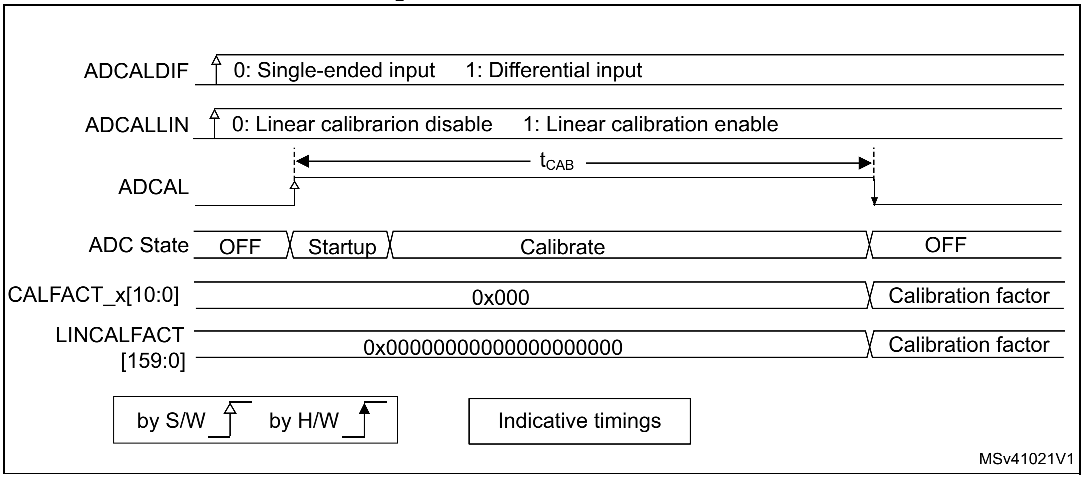
The calibration factor can be written if the ADC is enabled but not converting (ADEN=1 and ADSTART=0 and JADSTART=0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

Refer to the datasheets for the clock cycle requirement for both linear and offset calibration.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD=0, ADVREGEN=1 and verify that the ADC voltage regulator startup time has elapsed by checking the LDORDY bit in ADC_ISR (refer to *Section 28.3: ADC implementation* for the availability of the LDO regulator status).
2. Ensure that ADEN=0.
3. Select the input mode for this calibration by setting ADCALDIF=0 (single-ended input) or ADCALDIF=1 (Differential input). Select if Linearity calibration enable or not by ADCALLIN=1(enabled) or ADCALLIN=0(disabled).
4. Set ADCAL=1.
5. Wait until ADCAL=0.
6. The offset calibration factor can be read from ADC_CALFACT register.
7. The linearity calibration factor can be read from ADC_CALFACT2 register, following the procedure described in *Section : Linearity calibration reading procedure* (ADEN must be set to 1 prior to accessing ADC_CALFACT2 register).

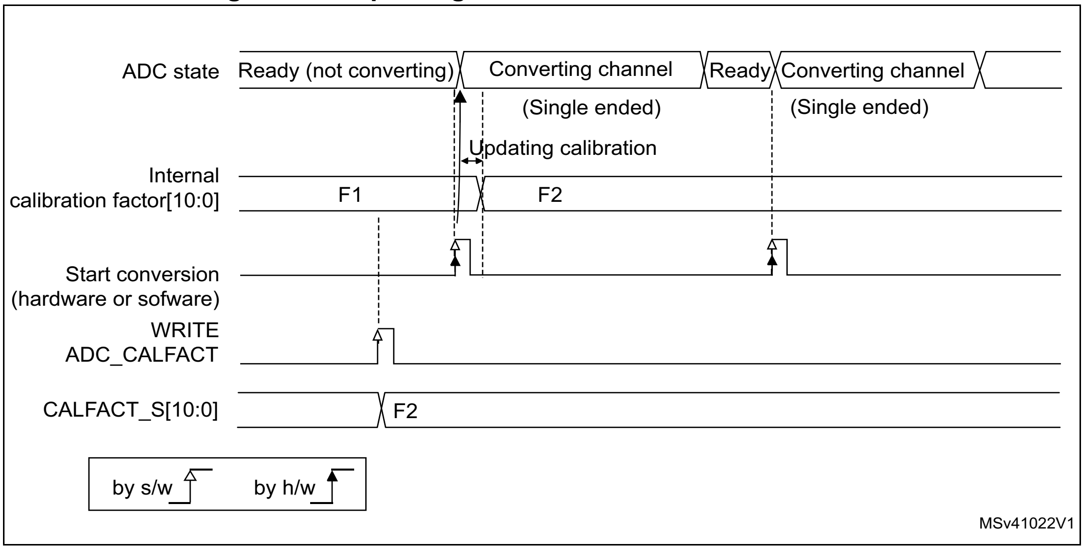
Figure 181. ADC calibration



Software procedure to re-inject a calibration factor into the ADC

1. Ensure ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new offset calibration factors.
3. Write LINCALFACT bits with the new linearity calibration factors, following the procedure described in *Section : Linearity calibration writing procedure*.
4. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 182. Updating the ADC offset calibration factor

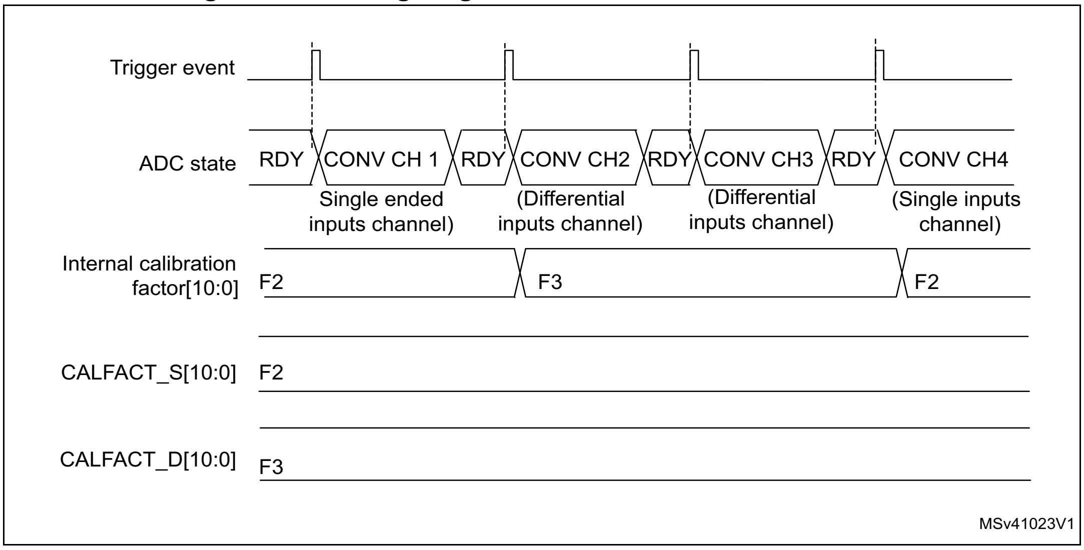


Calibrating single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF=0 and one with ADCALDIF=1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF=0) and Linearity calibration enable (with ADCALLIN=1). This updates the registers CALFACT_S[10:0] and LINCALFACT[159:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF=1) and Linearity calibration disable (with ADCALLIN=0). This updates the register CALFACT_D[10:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.

Figure 183. Mixing single-ended and differential channels



Linearity calibration reading procedure

Once the calibration is done (ADCAL bit cleared by hardware) with ADCALLIN=1, the 160-bit linearity correction factor can be read using the ADC_CALFACT2 30-bit registers (6 read accesses are necessary).

The six LINCALRDYW1..6 control/status bits in ADC_CR are set when the calibration is complete. When ADEN is set to 1, clearing one of these bits launches the transfer of part of the linearity factor into the LINCALFACT[29:0] of the ADC_CALFACT2 register. The bit is reset by hardware when the ADC_CALFACT2 register can be read (software must poll the bit until it is cleared). The complete procedure is as following:

1. Ensure DEEPPWD=0, ADVREGEN=1 and that the ADC voltage regulator startup time has elapsed by checking the LDORDY bit in ADC_ISR (refer to *Section 28.3: ADC implementation* for the availability of the LDO regulator status).
2. Set ADEN = 1 and wait until ADRDY=1.
3. Clear LINCALRDYW6 bit (Linearity calibration ready Word 6).
4. Poll LINCALRDYW6 bit until returned value is zero, indicating linearity correction bits[159:150] are available in ADC_CALFACT2[29:0].
5. Read ADC_CALFACT2[29:0].
6. Clear LINCALRDYW5 bit.
7. Poll LINCALRDYW5 bit until returned value is zero, indicating linearity correction bits[149:120] are available in ADC_CALFACT2[29:0].
8. Read ADC_CALFACT2[29:0].
9. Clear LINCALRDYW4 bit.
10. Poll LINCALRDYW4 bit until returned value is zero, indicating linearity correction bits[119:90] are available in ADC_CALFACT2[29:0].
11. Read ADC_CALFACT2[29:0].
12. Clear LINCALRDYW3 bit.
13. Poll LINCALRDYW3 bit until returned value is zero, indicating linearity correction bits[89:60] are available in ADC_CALFACT2[29:0].
14. Read ADC_CALFACT2[29:0].
15. Clear LINCALRDYW2 bit.
16. Poll LINCALRDYW2 bit until returned value is zero, indicating linearity correction bits[59:30] are available in ADC_CALFACT2[29:0].
17. Read ADC_CALFACT2[29:0].
18. Clear LINCALRDYW1 bit.
19. Poll LINCALRDYW1 bit until returned value is zero, indicating linearity correction bits[29:0] are available in ADC_CALFACT2[29:0].
20. Read ADC_CALFACT2[29:0].

Note: The software is allowed to toggle a single LINCALRDYWx bit at once (other bits left unchanged), otherwise causing unexpected behavior.

The software can access the linearity calibration factor by writing LINCALRDYW1..6 bits only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).

Linearity calibration writing procedure

The six LINCALRDYW1..6 control/status bits in ADC_CR are reset when the calibration has not yet been done or a new linearity calibration factor have been rewritten. It is possible to force directly a linearity calibration factor or re-inject it using the following procedure:

1. Ensure DEEPPWD=0, ADVREGEN=1 and that ADC voltage regulator startup time has elapsed by checking the LDORDY bit in ADC_ISR (refer to *Section 28.3: ADC implementation* for the availability of the LDO regulator status).
2. Set ADEN = 1 and wait until ADRDY=1.
3. Write ADC_CALFACT2[9:0] with previously saved linearity correction factor bits[159:150].
4. Set LINCALRDYW6 bit.
5. Poll LINCALRDYW6 bit until returned value is one, indicating linearity correction bits[159:150] have been effectively written.
6. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[149:120].
7. Set LINCALRDYW5 bit.
8. Poll LINCALRDYW5 bit until returned value is one, indicating linearity correction bits[149:120] have been effectively written.
9. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[119:90].
10. Set LINCALRDYW4 bit.
11. Poll LINCALRDYW4 bit until returned value is one, indicating linearity correction bits[119:90] have been effectively written.
12. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[89:60].
13. Set LINCALRDYW3 bit.
14. Poll LINCALRDYW3 bit until returned value is one, indicating linearity correction bits[89:60] have been effectively written.
15. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[59:30].
16. Set LINCALRDYW2 bit.
17. Poll LINCALRDYW2 bit until returned value is one, indicating linearity correction bits[59:30] have been effectively written.
18. Write ADC_CALFACT2[29:0] with previously saved linearity correction factor bits[29:0].
19. Set LINCALRDYW1 bit.
20. Poll LINCALRDYW1 bit until returned value is one, indicating linearity correction bits[29:0] have been effectively written.

Note: The software is allowed to toggle a single LINCALRDYWx bit at once (other bits left unchanged), otherwise causing unexpected behavior.

The software is allowed to update the linearity calibration factor by writing LINCALRDYW1..6 bits only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing).

28.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 28.4.6: ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 184*. Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART=1 (refer to *Section 28.4.19: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART=1 or when an external injected trigger event occurs, if injected triggers are enabled.

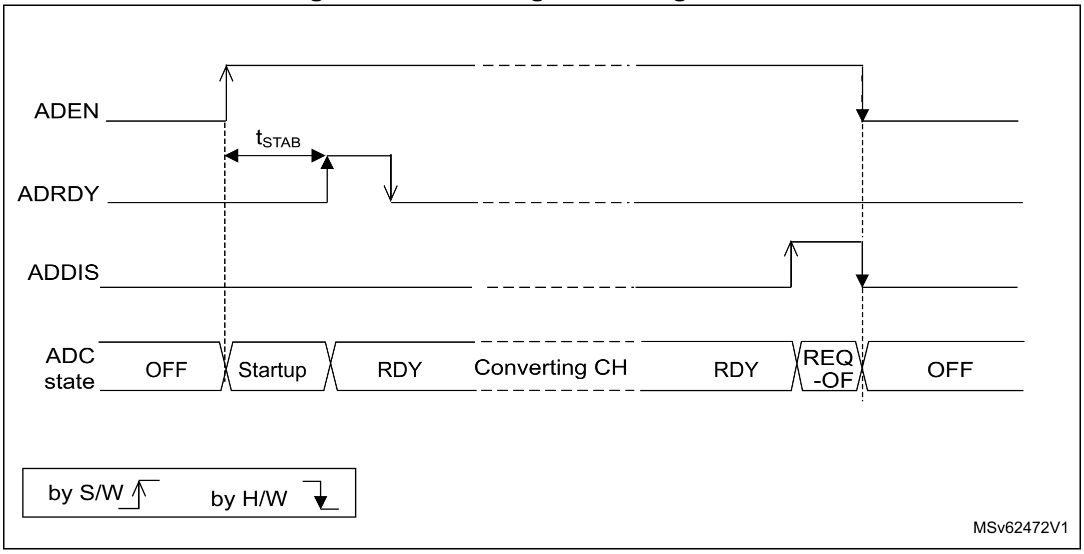
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN=1.
3. Wait until ADRDY=1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE=1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

Software procedure to disable the ADC

1. Check that both ADSTART=0 and JADSTART=0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP=1 and JADSTP=1 and then wait until ADSTP=0 and JADSTP=0.
2. Set ADDIS=1.
3. If required by the application, wait until ADEN=0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN=0).

Figure 184. Enabling / Disabling the ADC



28.4.10 Constraints when writing the ADC control bits

The software can write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the control bits DIFSEL in the ADC_DIFSEL register, ADC_CCR register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRy, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if there is no injected conversion ongoing (JADSTART must be equal to 0).

The software can write ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and eventually converting and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN=1).

The software is allowed to write the ADC_JSQR register only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN=0 as well as all the bits of ADC_CR register).

28.4.11 Channel selection (SQRx, JSQRx)

There are up to 20 multiplexed channels per ADC:

- 6 fast analog inputs coming from Analog PADs and GPIO pads (ADCx_INP/INN[0..5])
- Up to 14 slow analog inputs coming from GPIO pads (ADCx_INP/INN[6..19]).
- The ADCs are connected to 6 internal analog inputs:
  - the internal temperature sensor (V_{SENSE})
  - the internal reference voltage (V_{REFINT})
  - the V_{DDCORE} monitoring channel
  - the V_{BAT} monitoring channel ($V_{BAT}/4$)
  - DAC internal channels

Refer to *Table ADC interconnection* in *Section 28.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to external ADC pins or internal signals.

It is possible to organize the conversions in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INP/INN3, ADCx_INP/INN8, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN0, ADCx_INP/INN2, ADCx_INP/INN2, ADCx_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP=1 (refer to *Section 28.4.18: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 only when the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Temperature sensor, V_{REFINT}, V_{DDCORE} and V_{BAT} internal channels

The internal reference voltage (V_{REFINT}), the temperature sensor (V_{SENSE}), V_{DDCORE} monitoring channel, and the V_{BAT} channel are connected to ADC internal channels. Refer to *Table ADC interconnection* in *Section 28.4.2: ADC pins and internal signals* for details.

Note: To convert one of the internal analog channels, enable the corresponding analog sources by programming VREFEN, TSEN and VBATEN bits in the ADC_CCR registers, or VDDCOREEN bit in the ADC2_OR register.

28.4.12 Channel preselection register (ADC_PCSEL)

For each channel selected through SQRx or JSQRx, the corresponding ADC_PCSEL bit must be previously configured.

This ADC_PCSEL bit controls the analog switch integrated in the I/O level. The ADC input MUX selects the ADC input according to the SQRx and JSQRx with very high speed, the analog switch integrated in the IO cannot react as fast as ADC mux does. To avoid the delay on analog switch control on IO, it is necessary to preselect the input channels which are selected in the SQRx, JSQRx.

The selection is based on the $V_{INP[i]}$ of each ADC input. If ADC1 converts the ADC1_INP2($V_{INP[2]}$) as differential mode, ADC1_INP6($V_{INN[2]}$) also needs to be selected in ADC_PCSEL.

Some I/Os are connected to several $V_{INP[i]}$ of the ADCx. The control inputs of the analog switch are ORed with the corresponding ADC_PCSEL register bits.

28.4.13 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 8.5 ADC clock cycles
- SMP = 011: 16.5 ADC clock cycles
- SMP = 100: 32.5 ADC clock cycles
- SMP = 101: 64.5 ADC clock cycles
- SMP = 110: 387.5 ADC clock cycles
- SMP = 111: 810.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 7.5 ADC clock cycles

Example:

With $F_{adc\_ker\_ck}$ = 24 MHz and a sampling time of 1.5 ADC clock cycles (14-bit mode):

T_{CONV} = (1.5 + 7.5) ADC clock cycles = 9 ADC clock cycles = 0.375 µs (14 bit mode for fast channels)

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Constraints on the sampling time for fast and slow channels

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

I/O analog switch voltage booster

The resistance of the I/O analog switches increases when the V_{DDA} voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low V_{DDA} voltage by enabling an internal voltage booster through the EN_BOOSTER bit of the SYSCFG_PMCR register.

28.4.14 Single conversion mode (CONT=0)

In single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel, with software trigger selected)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel, with software trigger selected)
- External hardware trigger event (for a regular or injected channel)

  ADSTART bit or JADSTART bit must be set before triggering an external event.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

28.4.15 Continuous conversion mode (CONT=1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically re-starts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).

28.4.16 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART=1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN /= 0x0

Software starts ADC injected conversions by setting JADSTART=1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN /= 0x0

Note: In auto-injection mode (JAUTO=1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART=0 and JADSTART=0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software trigger (CONT=0, EXTEN=0x0)
  - at any end of conversion sequence (EOS =1)
- In discontinuous mode with software trigger (CONT=0, DISCEN=1, EXTEN=0x0)
  - at end of conversion (EOC=1)
- In all other cases (CONT=x, EXTEN=x)
  - after execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT=1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT=0 and EXTEN /=0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- in single mode with software injected trigger (JEXTEN=0x0)
  - at any end of injected conversion sequence (JEOS assertion) or at any end of sub-group processing if JDISCEN=1
- in all cases (JEXTEN=x)
  - after execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

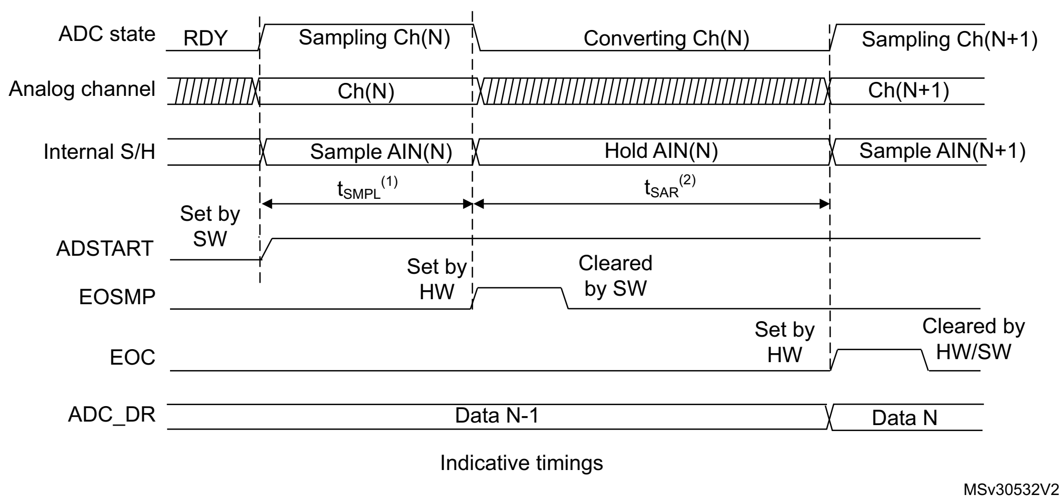
28.4.17 Timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$$T_{CONV} = T_{SMPL} + T_{SAR} = [1.5\ _{|min} + 7.5\ _{|14bit}] \times T_{adc\_ker\_ck}$$

$$T_{CONV} = T_{SMPL} + T_{SAR} = 62.5\text{ ns}\ _{|min} + 312.5\text{ ns}\ _{|14bit} = 375.0\text{ ns (for } F_{adc\_ker\_ck} = 24\text{ MHz)}$$

Figure 185. Analog-to-digital conversion time



1. t_{SMPL} depends on SMP[2:0]
2. t_{SAR} depends on RES[2:0]

28.4.18 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP=1 and injected conversions ongoing by setting JADSTP=1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would re-start a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 186. Stopping ongoing regular conversions

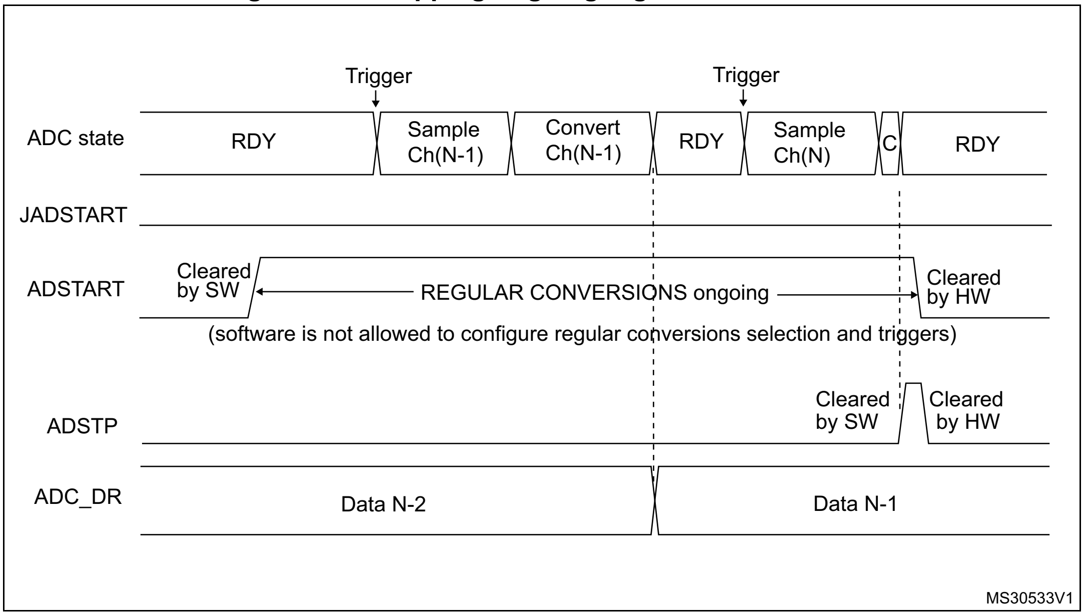
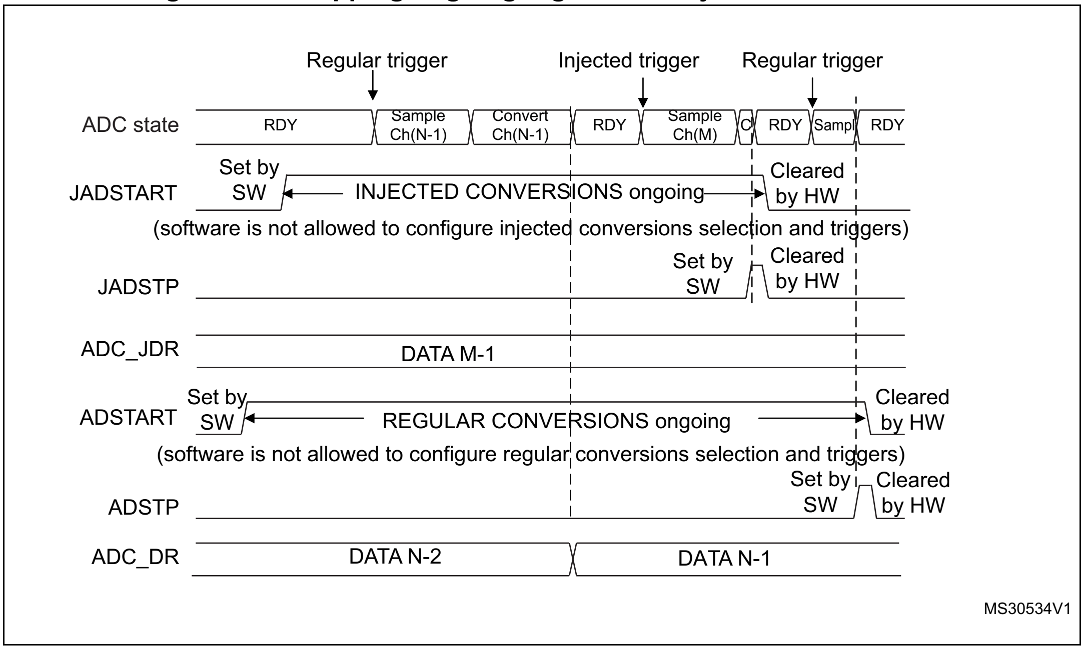


Figure 187. Stopping ongoing regular and injected conversions



28.4.19 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (e.g. timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS=0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART=1 and the injected trigger selection is effective once software has set bit JADSTART=1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART=0, any regular hardware triggers which occur are ignored.
- If bit JADSTART=0, any injected hardware triggers which occur are ignored.

Table 188 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 188. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 189. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	– If JQDIS=1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS=0 (Queue enabled), Hardware and software trigger detection disabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS=0). Refer to Section 28.4.22: Queue of context for injected conversions.

The EXTSEL[4:0] and JEXTSEL[4:0] control bits select which out of 21 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly.
The injected trigger selection can be anticipated and changed on-the-fly. Refer to *Section 28.4.22: Queue of context for injected conversions on page 1513*

Each ADC master shares the same input triggers with its ADC slave as described in *Figure 188*.

Figure 188. Triggers are shared between ADC master and ADC slave

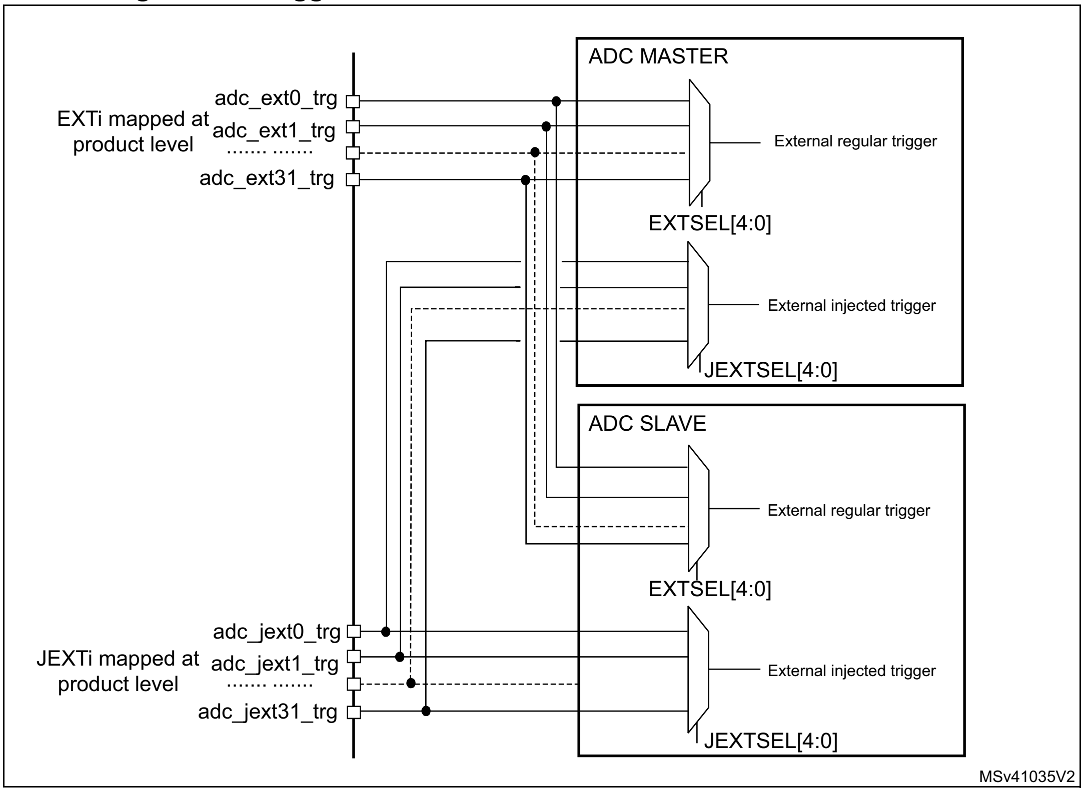


Table 190 and *Table 191* give all the possible external triggers of the three ADCs for regular and injected conversion.

Table 190. ADC1, ADC2 - External triggers for regular channels

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg0	tim1_oc1	Internal signal from on-chip timers	00000
adc_ext_trg1	tim1_oc2	Internal signal from on-chip timers	00001
adc_ext_trg2	tim1_oc3	Internal signal from on-chip timers	00010
adc_ext_trg3	tim2_oc2	Internal signal from on-chip timers	00011
adc_ext_trg4	tim3_trgo	Internal signal from on-chip timers	00100
adc_ext_trg5	tim4_oc4	Internal signal from on-chip timers	00101
adc_ext_trg6	exti11	External pin	00110
adc_ext_trg7	tim8_trgo	Internal signal from on-chip timers	00111
adc_ext_trg8	tim8_trgo2	Internal signal from on-chip timers	01000
adc_ext_trg9	tim1_trgo	Internal signal from on-chip timers	01001

Table 190. ADC1, ADC2 - External triggers for regular channels (continued)

Name	Source	Type	EXTSEL[4:0]
adc_ext_trg10	tim1_trgo2	Internal signal from on-chip timers	01010
adc_ext_trg11	tim2_trgo	Internal signal from on-chip timers	01011
adc_ext_trg12	tim4_trgo	Internal signal from on-chip timers	01100
adc_ext_trg13	tim6_trgo	Internal signal from on-chip timers	01101
adc_ext_trg14	tim15_trgo	Internal signal from on-chip timers	01110
adc_ext_trg15	tim3_oc4	Internal signal from on-chip timers	01111
adc_ext_trg16	Reserved	Internal signal from on-chip timers	10000
adc_ext_trg17	Reserved	Internal signal from on-chip timers	10001
adc_ext_trg18	lptim1_out	Internal signal from on-chip timers	10010
adc_ext_trg19	lptim2_out	Internal signal from on-chip timers	10011
adc_ext_trg20	lptim3_out	Internal signal from on-chip timers	10100
adc_ext_trg21	Reserved	-	10101
adc_ext_trg22	Reserved	-	10110
adc_ext_trg23	Reserved	-	10111
adc_ext_trg24	Reserved	-	11000
adc_ext_trg25	Reserved	-	11001
adc_ext_trg26	Reserved	-	11010
adc_ext_trg27	Reserved	-	11011
adc_ext_trg28	Reserved	-	11100
adc_ext_trg29	Reserved	-	11101
adc_ext_trg30	Reserved	-	11110
adc_ext_trg31	Reserved	-	11111

Table 191. ADC1, ADC2- External triggers for injected channels

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg0	tim1_trgo	Internal signal from on-chip timers	00000
adc_jext_trg1	tim1_oc4	Internal signal from on-chip timers	00001
adc_jext_trg2	tim2_trgo	Internal signal from on-chip timers	00010
adc_jext_trg3	tim2_oc1	Internal signal from on-chip timers	00011
adc_jext_trg4	tim3_oc4	Internal signal from on-chip timers	00100
adc_jext_trg5	tim4_trgo	Internal signal from on-chip timers	00101
adc_jext_trg6	exti15	External pin	00110
adc_jext_trg7	tim8_oc4	Internal signal from on-chip timers	00111
adc_jext_trg8	tim1_trgo2	Internal signal from on-chip timers	01000
adc_jext_trg9	tim8_trgo	Internal signal from on-chip timers	01001

Table 191. ADC1, ADC2- External triggers for injected channels (continued)

Name	Source	Type	JEXTSEL[4:0]
adc_jext_trg10	tim8_trgo2	Internal signal from on-chip timers	01010
adc_jext_trg11	tim3_oc3	Internal signal from on-chip timers	01011
adc_jext_trg12	tim3_trgo	Internal signal from on-chip timers	01100
adc_jext_trg13	tim3_oc1	Internal signal from on-chip timers	01101
adc_jext_trg14	tim6_trgo	Internal signal from on-chip timers	01110
adc_jext_trg15	tim15_trgo	Internal signal from on-chip timers	01111
adc_jext_trg16	Reserved	Internal signal from on-chip timers	10000
adc_jext_trg17	Reserved	Internal signal from on-chip timers	10001
adc_jext_trg18	lptim1_out	Internal signal from on-chip timers	10010
adc_jext_trg19	lptim2_out	Internal signal from on-chip timers	10011
adc_jext_trg20	lptim3_out	Internal signal from on-chip timers	10100
adc_jext_trg21	Reserved	-	10101
adc_jext_trg22	Reserved	-	10110
adc_jext_trg23	Reserved	-	10111
adc_jext_trg24	Reserved	-	11000
adc_jext_trg25	Reserved	-	11001
adc_jext_trg26	Reserved	-	11010
adc_jext_trg27	Reserved	-	11011
adc_jext_trg28	Reserved	-	11100
adc_jext_trg29	Reserved	-	11101
adc_jext_trg30	Reserved	-	11110
adc_jext_trg31	Reserved	-	11111

28.4.20 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is

reset and the injected channel sequence switches are launched (all the injected channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 189* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 20 ADC clock cycles (that is two conversions with a sampling time of 1.5 clock periods), the minimum interval between triggers must be 21 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

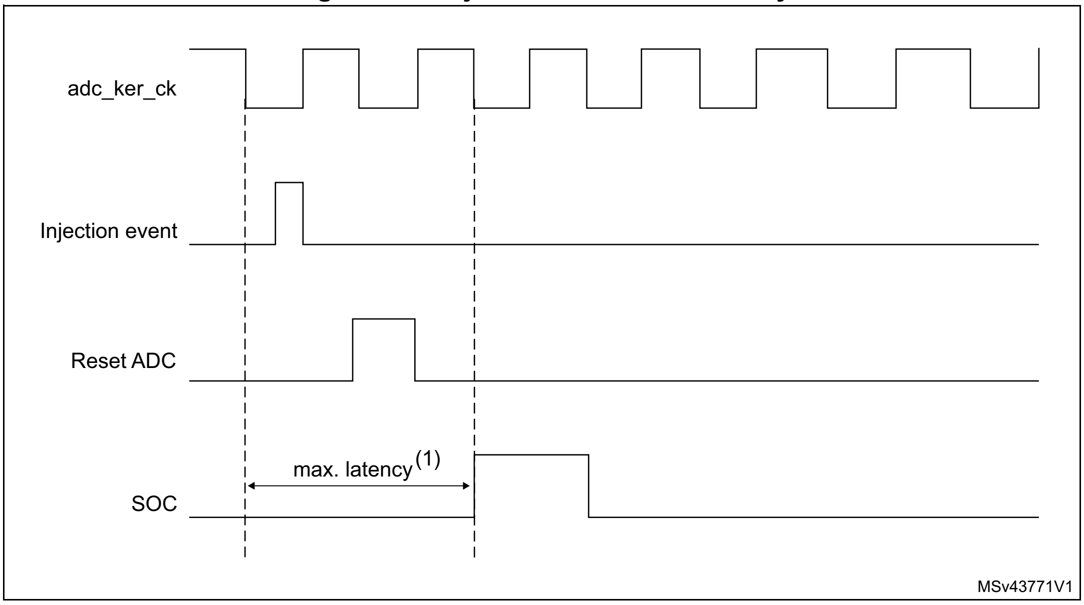
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode. If the single-shot mode is selected, the JAUTO sequence is stopped upon DMA Transfer Complete event.

Figure 189. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

28.4.21 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (sub-group) of n conversions ($n \leq 8$) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQR registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN=1, n=3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN=0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - all the next trigger events relaunch the complete sequence.

Note: When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN=1, CONT=1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where 'n' is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN=1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

28.4.22 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL[4:0] in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The ADC_JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the ADC_JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the ADC_JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next ADC_JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into ADC_JSQR register while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of ADC_JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM=0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM=1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers

are ignored until the software re-writes a new injected context into ADC_JSQR register.

- Reading ADC_JSQR register returns the current ADC_JSQR context which is active at that moment. When the ADC_JSQR context is empty, JSQi is read as 0x00.
- The Queue is flushed when stopping injected conversions by setting JADSTP=1 or when disabling the ADC by setting ADDIS=1:
  - If JQM=0, the Queue is maintained with the last active context.
  - If JQM=1, the Queue becomes empty and triggers are ignored.

Note: When configured in discontinuous mode (bit JDISCEN=1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

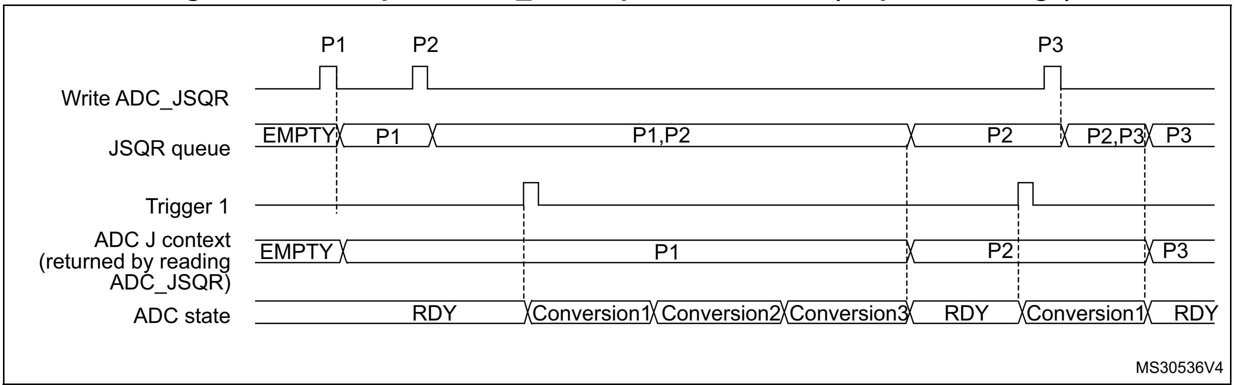
- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

Note: When queue of context enabled (bit JQDIS=0), only hardware trigger can be used.

Behavior when changing the trigger or sequence context

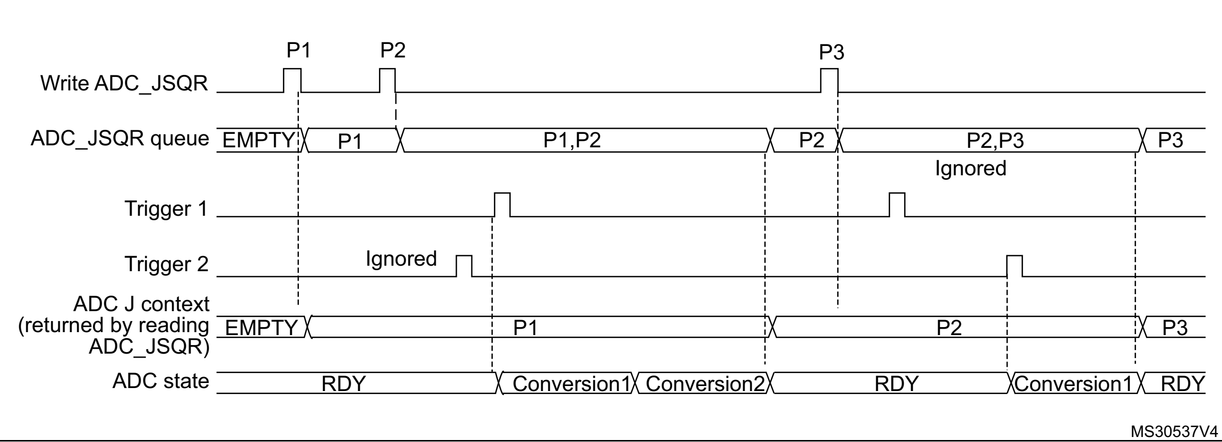
The *Figure 190* and *Figure 191* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 190. Example of ADC_JSQR queue of context (sequence change)



1. Parameters:
P1: sequence of 3 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 4 conversions, hardware trigger 1

Figure 191. Example of ADC_JSQR queue of context (trigger change)

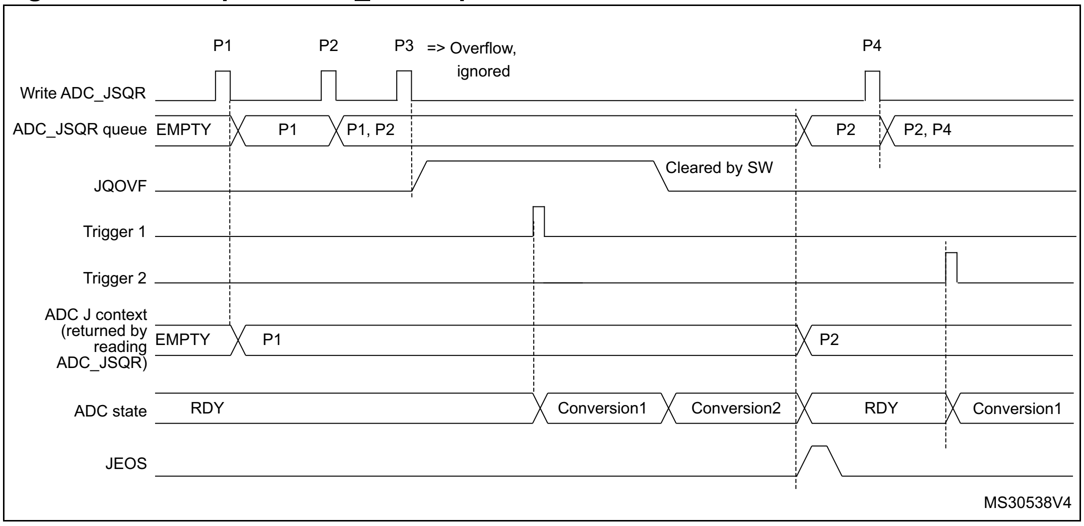


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

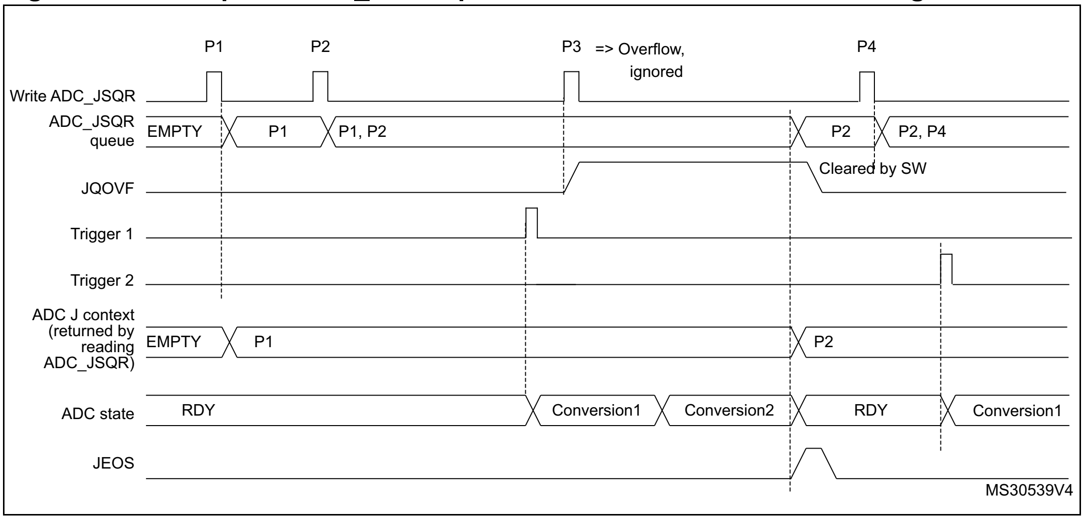
The *Figure 192* and *Figure 193* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 192. Example of ADC_JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 193. Example of ADC_JSQR queue of context with overflow during conversion



1. Parameters:
P1: sequence of 2 conversions, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 2
P3: sequence of 3 conversions, hardware trigger 1
P4: sequence of 4 conversions, hardware trigger 1

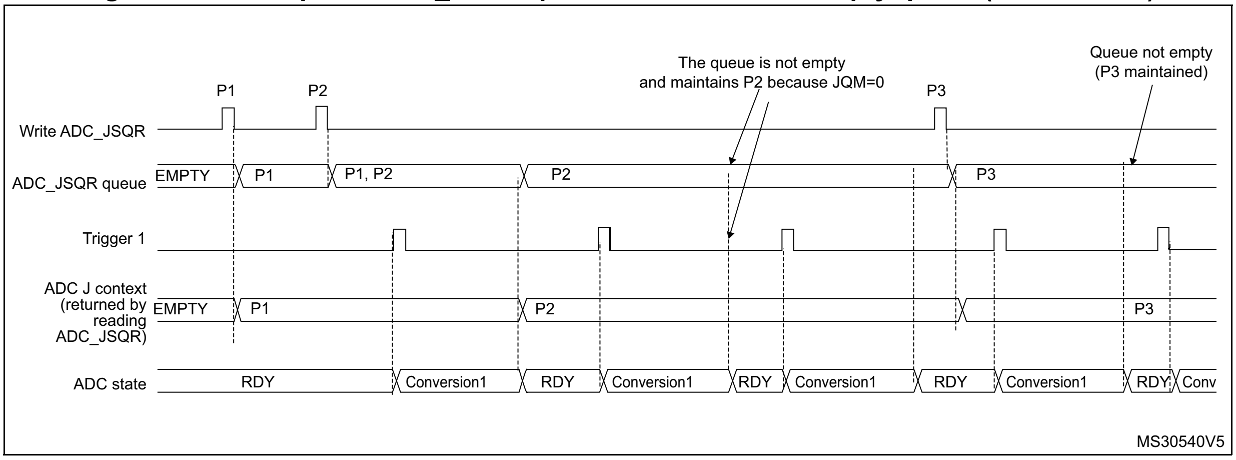
It is recommended to manage the queue overflows as described below:

- After each P context write into ADC_JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 194 and *Figure 195* show the behavior of the context Queue when the Queue becomes empty in both cases JQM=0 or 1.

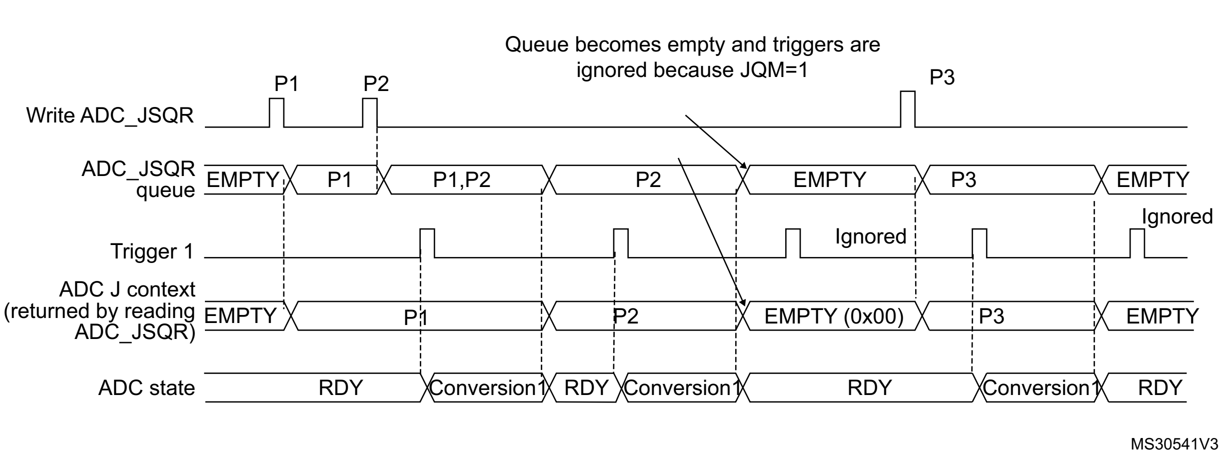
Figure 194. Example of ADC_JSQR queue of context with empty queue (case JQM=0)



1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 195. Example of ADC_JSQR queue of context with empty queue (case JQM=1)

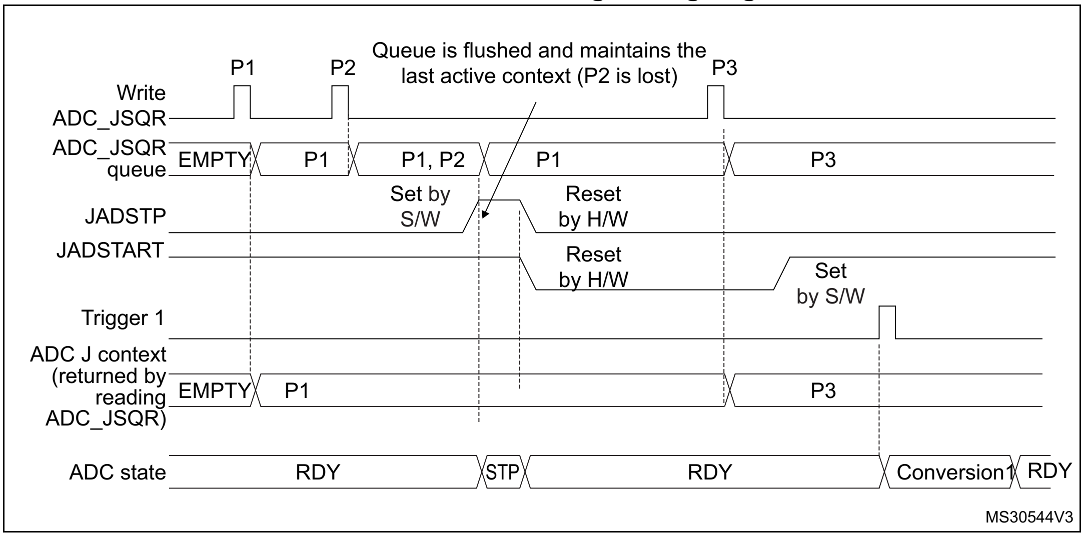


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

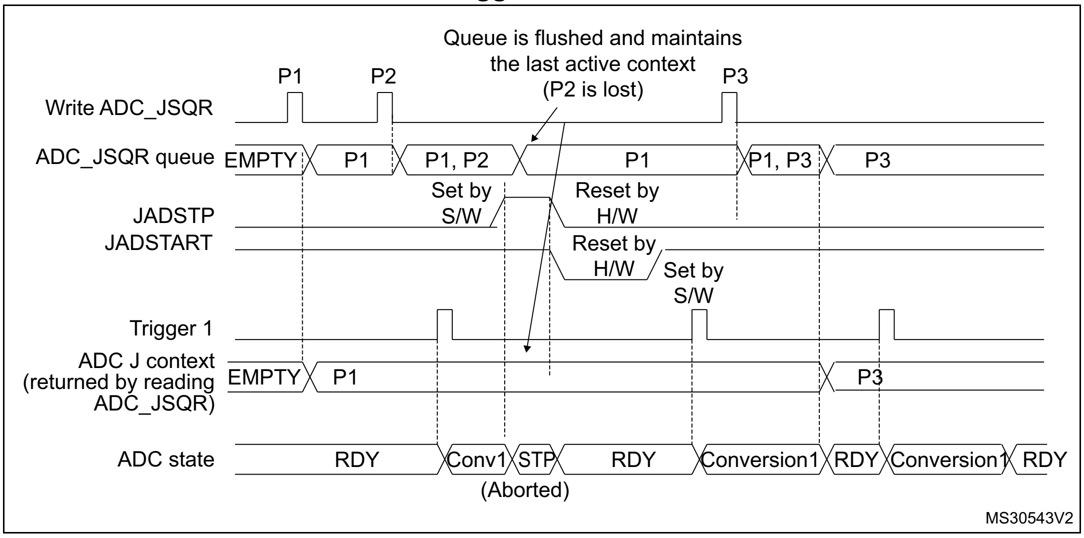
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 196. Flushing ADC_JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion.



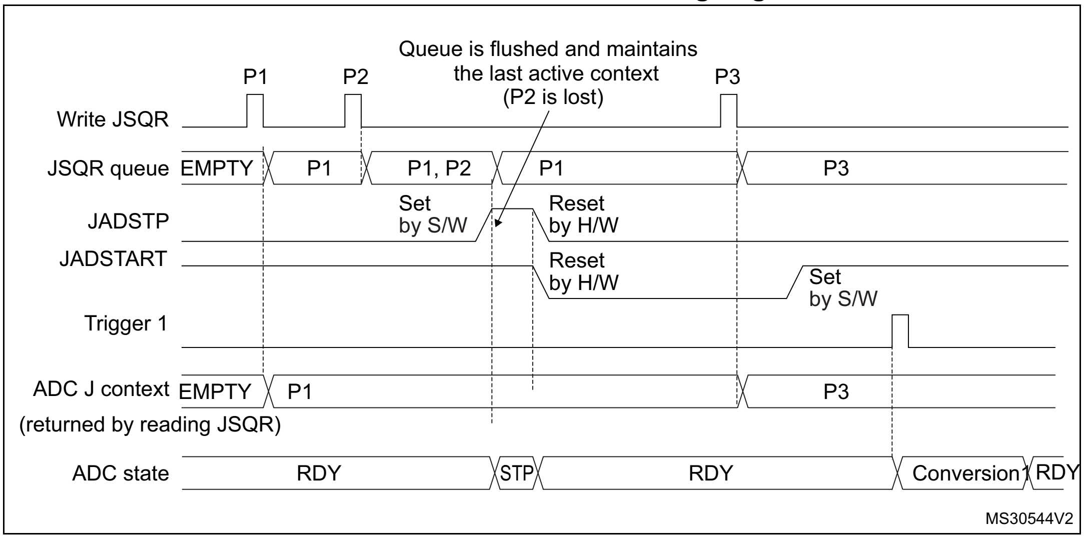
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 197. Flushing ADC_JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs during an ongoing conversion and a new trigger occurs.



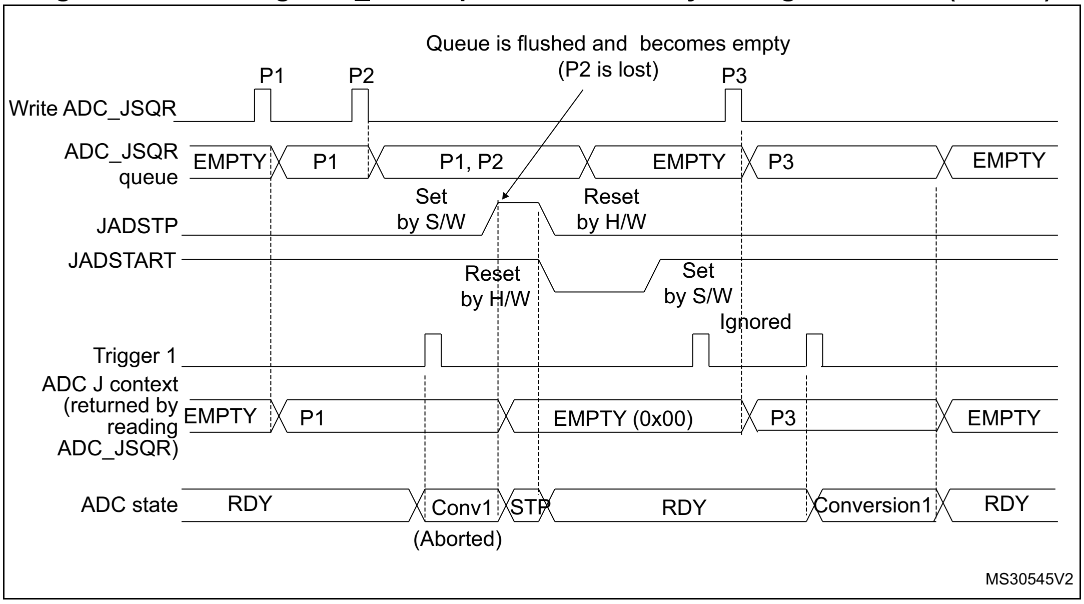
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 198. Flushing ADC_JSQR queue of context by setting JADSTP=1 (JQM=0). Case when JADSTP occurs outside an ongoing conversion



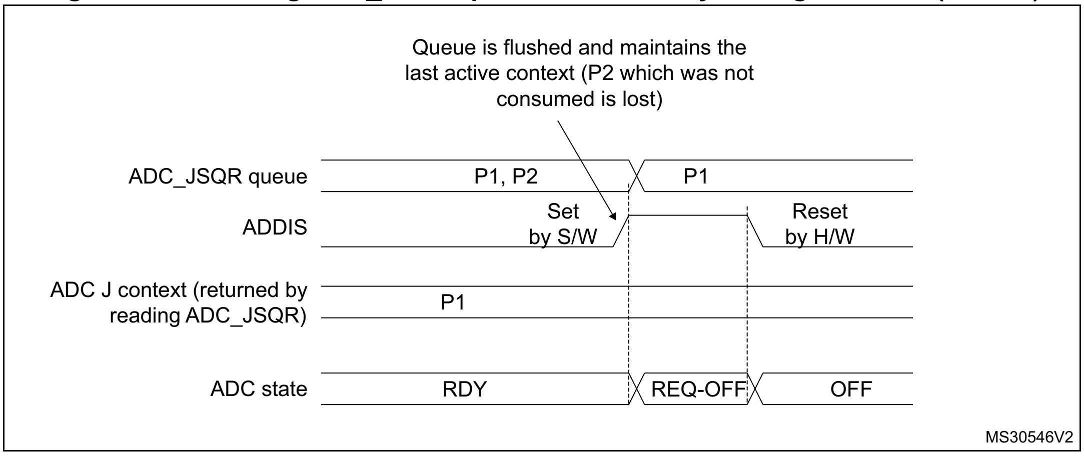
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 199. Flushing ADC_JSQR queue of context by setting JADSTP=1 (JQM=1)



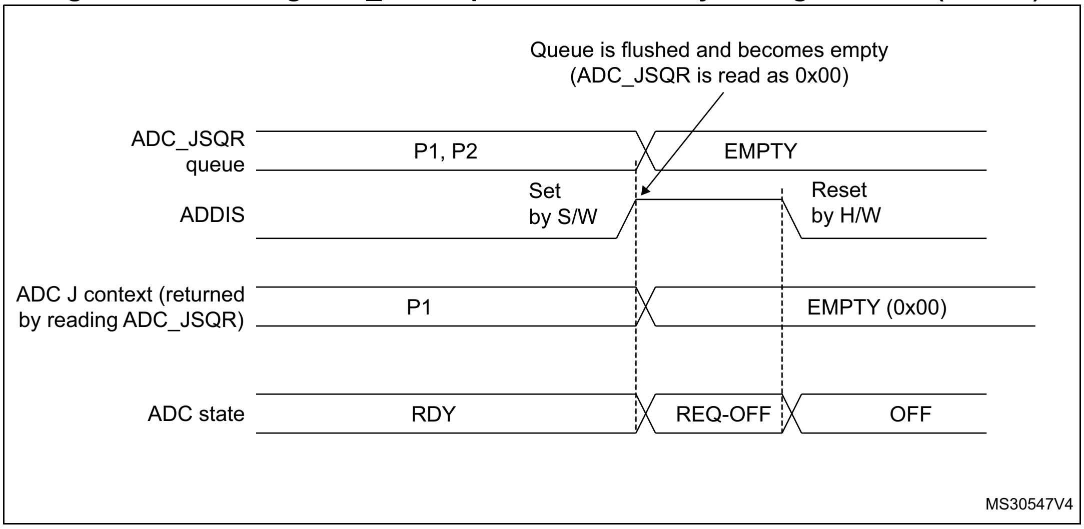
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 200. Flushing ADC_JSQR queue of context by setting ADDIS=1 (JQM=0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 201. Flushing ADC_JSQR queue of context by setting ADDIS=1 (JQM=1)



1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy ADC_JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS=1 into the ADC_CFGR register.

Queue of context: Programming of the register ADC_JSQR

When the injected conversion queue of context is enabled (JQDIS=0), the ADC_JSQR must be programmed at one register write access. As JL[1:0] register define the number of the injected sequence, corresponding JSQ1 to JSQ4 must be written at same time. If ADC_JSQR is reprogrammed before the injected conversion start, reprogrammed data is put on the queue. When queue of context is empty, ADC_JSQR read back as 0x0000. Register access should not use the ‘read modify write’ sequence.

When ADC_JSQR is programmed when already two contexts are queued, it raises JQOVF flag and generate the interrupt.

28.4.23 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 16, 14, 12, 10, 8 bits by programming the control bits RES[1:0]. *Figure 206*, *Figure 207*, *Figure 208* and *Figure 209* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 192*.

Table 192. T_{SAR} timings depending on resolution

RES [2:0]	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc\_ker\_ck}$=24 MHz	$T_{adc\_ker\_ck}$ (ADC clock cycles) (with Sampling Time= 1.5 ADC clock cycles)	$T_{adc\_ker\_ck}$ (ns) at $F_{adc\_ker\_ck}$=24 MHz
16 bits	8.5 ADC clock cycles	354.2	10 ADC clock cycles	416.7
14 bits	7.5 ADC clock cycles	312.5	9 ADC clock cycles	375
12 bits	6.5 ADC clock cycles	270.8	8 ADC clock cycles	333.3
10 bits	5.5 ADC clock cycles	229.2	7 ADC clock cycles	291.7
8 bits	4.5 ADC clock cycles	187.5	6 ADC clock cycles	250.0

28.4.24 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

28.4.25 End of conversion sequence (EOS, JEOS)

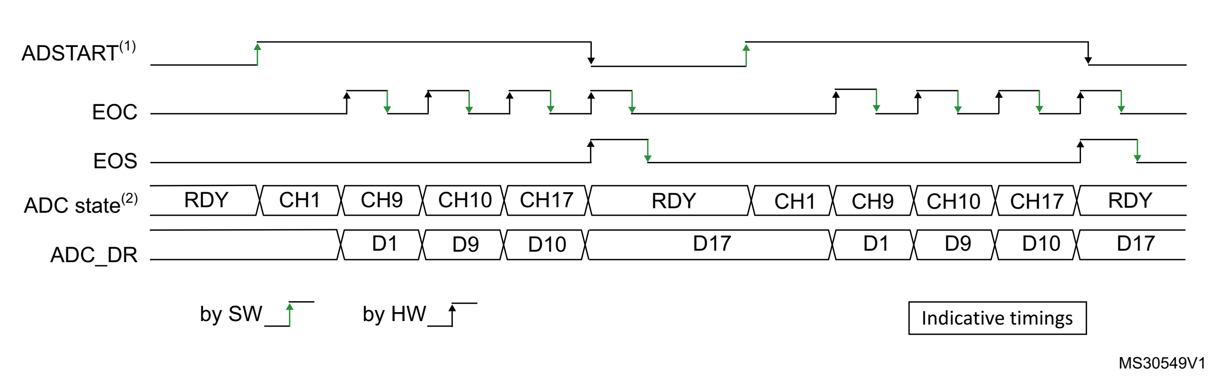
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

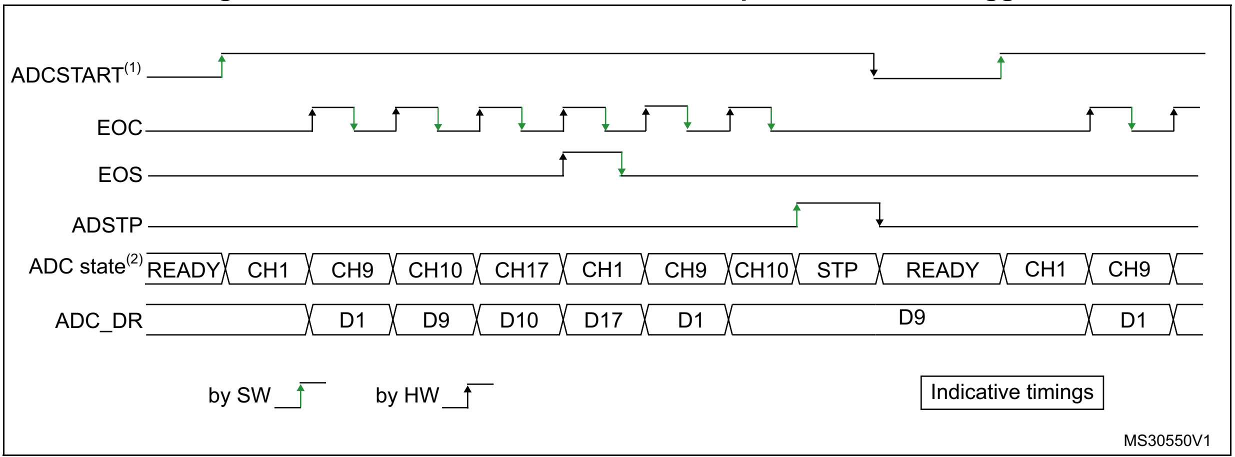
28.4.26 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 202. Single conversions of a sequence, software trigger



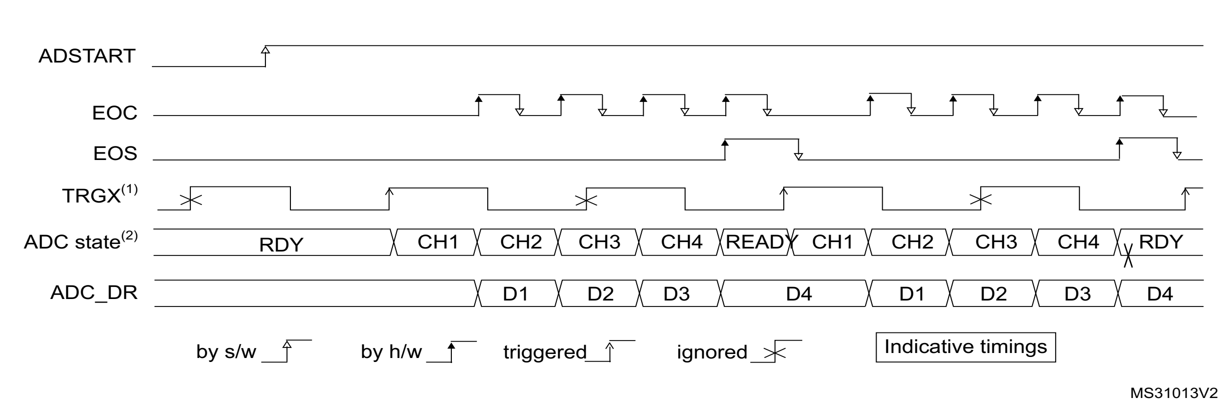
1. EXTEN=0x0, CONT=0
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 203. Continuous conversion of a sequence, software trigger



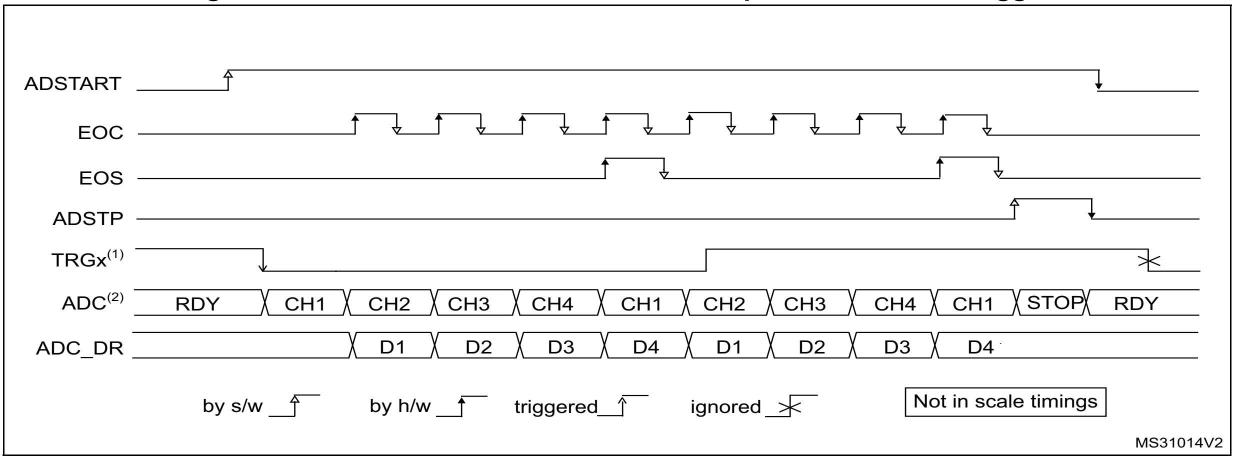
1. EXTEN=0x0, CONT=1
2. Channels selected = 1,9, 10, 17; AUTDLY=0.

Figure 204. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

Figure 205. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY=0.

28.4.27 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 32 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 32 bits wide.

The OVSS[3:0] and LSHIFT[3:0] bitfields in the ADC_CFGR2 register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 206*, *Figure 207*, *Figure 208* and *Figure 209*.

Note: The data can be re-aligned in normal and in oversampling mode.

Offset

An offset y (y=1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in OFFSETy[25:0] bitfield into ADC_OFRy register. The channel to which the offset is applied is programmed into the bits OFFSETy_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSETy[25:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

The offset value should be lower than the max conversion value (ex. 16bit mode, offset value max is 0xFFFF).

The offset correction is also supported in oversampling mode. For the oversampling mode, offset is subtracted before OVSS right shift applied.

Table 193 describes how the comparison is performed for all the possible resolutions for analog watchdog 1, 2, 3.

Table 193. Offset computation versus data resolution

Resolution (bits RES[2:0])	Subtraction between raw converted data and offset:		Result	Comments
	Raw converted Data, left aligned	Offset		
16 bits	DATA[15:0]	OFFSET[25:0]	signed 27-bit data	-
14 bits	DATA[15:2],00	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[1:0] to 00
12 bits	DATA[15:4],00 00	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[3:0] to 0000
10 bits	DATA[15:6],00 0000	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[5:0] to 000000
8 bits	DATA[15:8],00 0000	OFFSET[25:0]	signed 27-bit data	The user must configure OFFSET[7:0] to 00000000

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y=1,2,3,4) corresponding to channel i:

- If one of the offsets is enabled (bit OFFSETy_EN=1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 206, *Figure 207*, *Figure 208* and *Figure 209* show alignments for signed and unsigned data together with corresponding OVSS and LSHIFT values.

Figure 206. Right alignment (offset disabled, unsigned value)

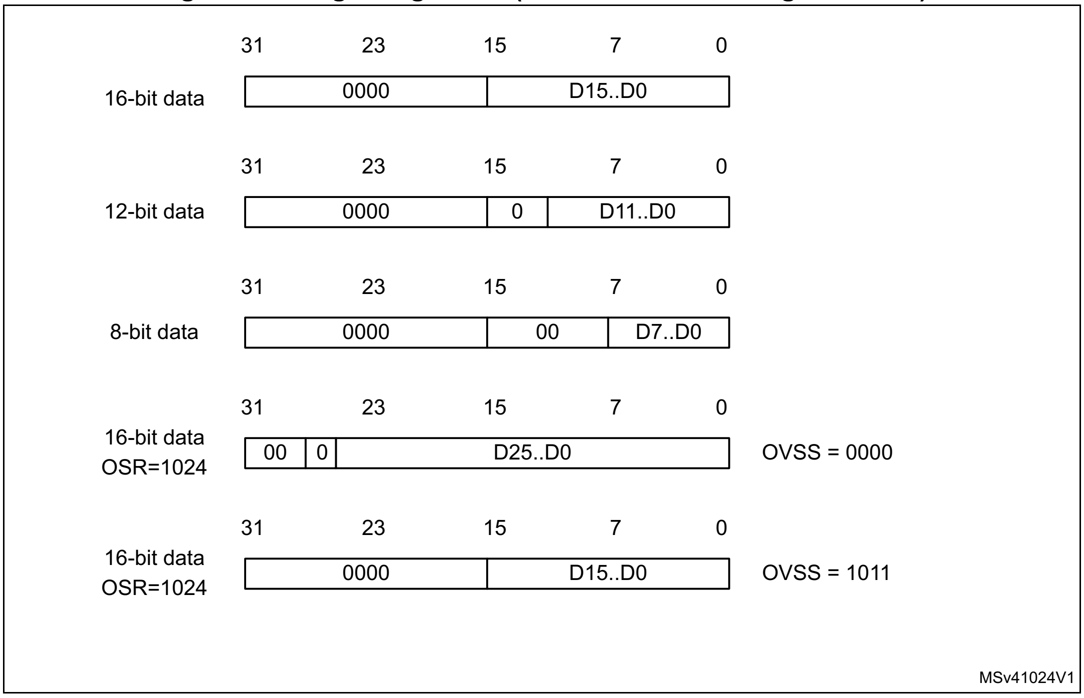


Figure 207. Right alignment (offset enabled, signed value)

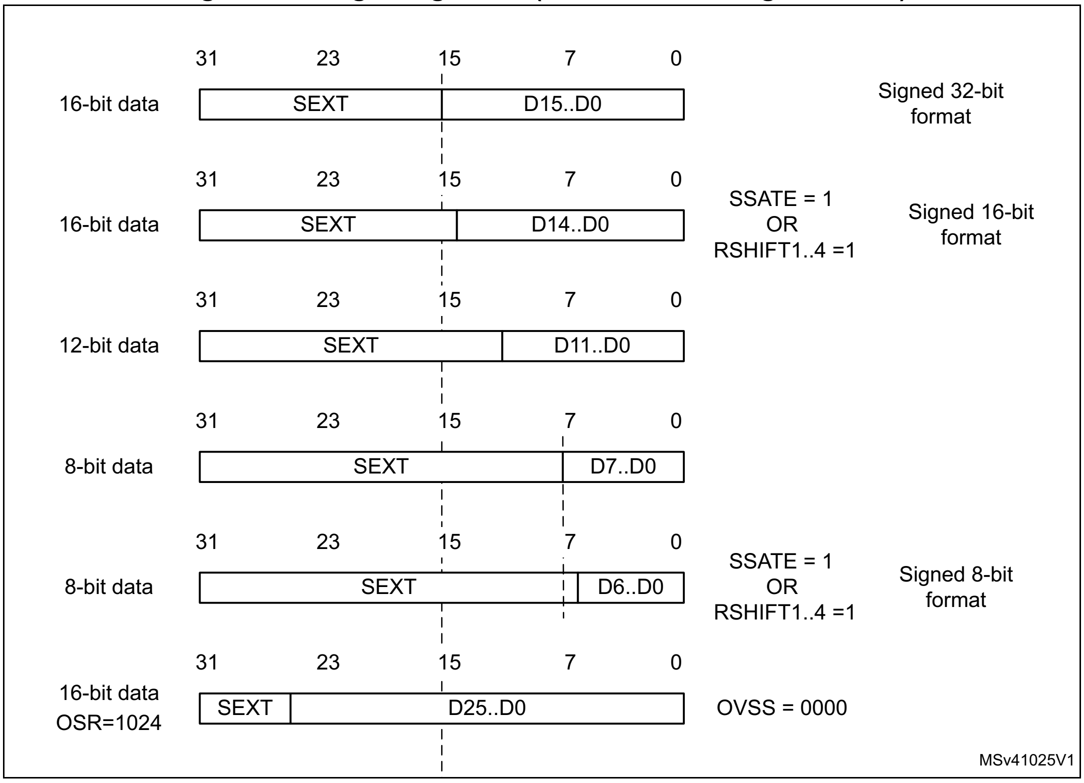


Figure 208. Left alignment (offset disabled, unsigned value)

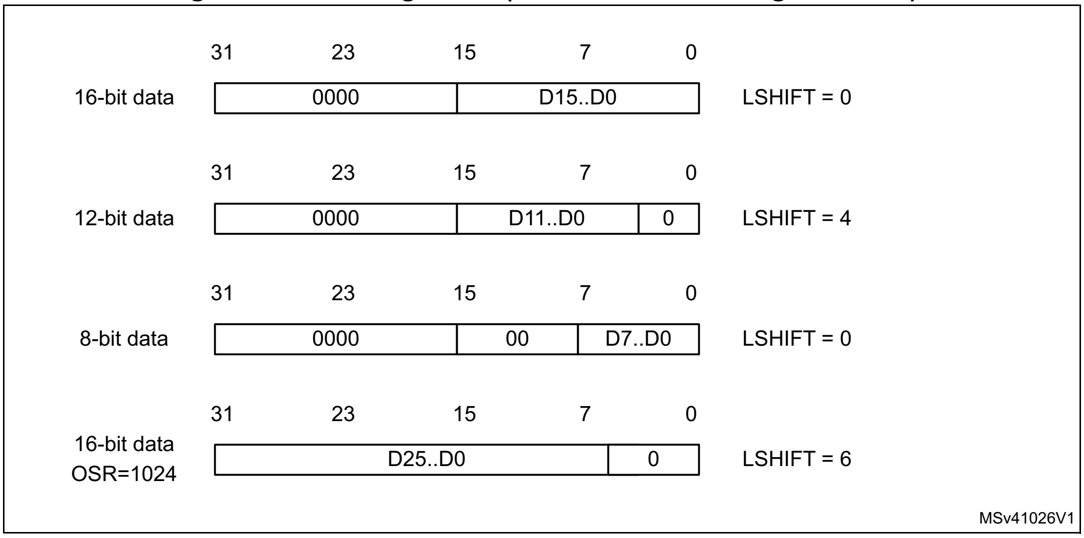
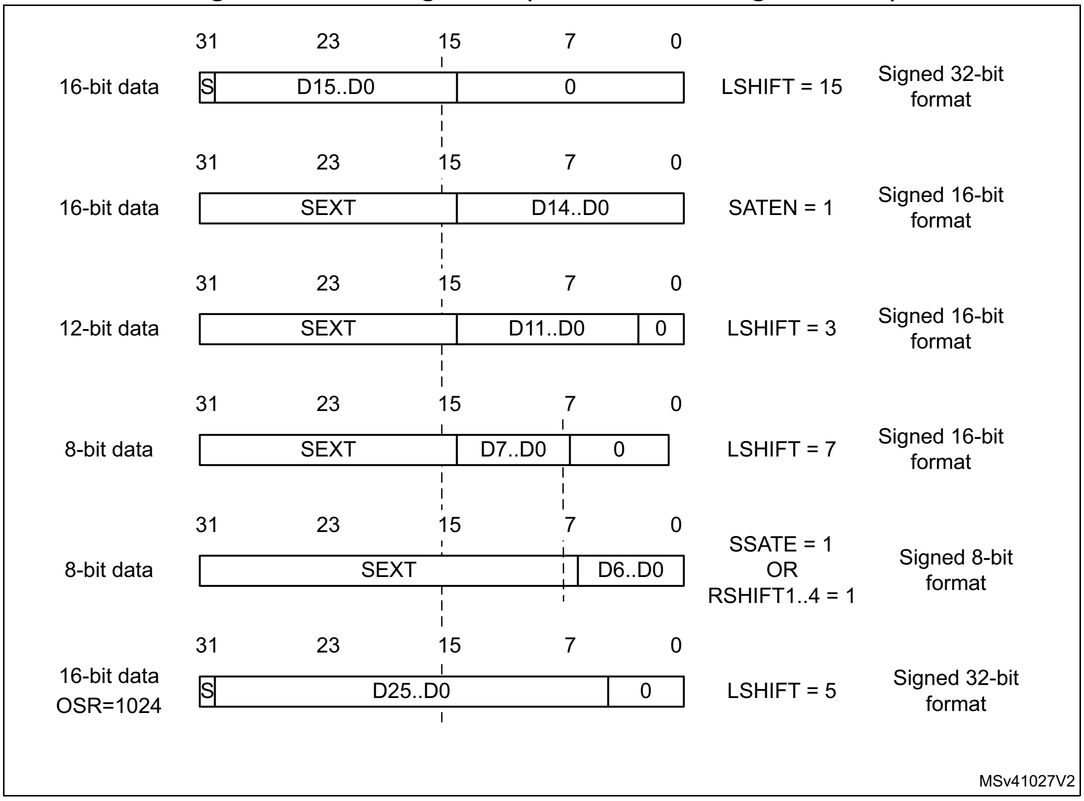


Figure 209. Left alignment (offset enabled, signed value)



16-bit and 8-bit signed format management: RSHIFTx,SSATE

The offset correction sign-extends the data format, resulting in an unsigned 16-bit conversion being extended to 17-bit signed format, for instance.

Three options are offered for formatting 8-bit and 16-bit conversion results.

For each offset correction channel 1 to 4, a RSHIFT1..4 bit in the ADC_CFGR2 register allows to have the result right-shifted 1-bit and have it fitting a standard 8 or 16-bit format.

Another option is to have the result saturated to the 16-bit and 8-bit signed formats, for the following cases only: RES[2:0] = 000 (16-bit format) and RES[2:0] = 100 (8-bit format).

This mode is enabled with the SSATE bit in the ADC_OFRy register.

The table below summarizes the 3 available use case for 16-bit format.

Table 194. 16-bit data formats

SSATE	RSHIFTx	Format	Data range (offset = 0x8000)
0	0	Sign-extended 17-bit significant data SEXT[31:16] DATA[15:0]	0x00007FFF - 0x FFFF8000
0	1	Sign-extended right-shifted 16-bit significant data SEXT[31:15] DATA[14:0]	0x3FFF - 0xC000
1	0	Sign-extended saturated 16-bit significant data SEXT[31:15] DATA[14:0]	7FFF - 0x8000
1	1	Reserved	-

Numerical examples are given in *Table 195* with 3 different offset values.

Table 195. Numerical examples for 16-bit format (bold indicates saturation)

Raw conversion result	Offset value	Result SSATE = 0 RSHIFT = 0	Result SSATE = 0 RSHIFT = 1	Result SSATE = 1 RSHIFT = 0
0xFFFF	0x8000	0x0000 7FFF	3FFF	7FFF
0x8000		0x0000 0000	0	0
0x0000		0xFFFF 8000	C000	8000
0xFFFF	0x8020	0x0000 7FDF	3FEF	7FDF
0x8000		0xFFFF FFE0	FFF0	FFE0
0x0000		0xFFFF 7FE0	BFF0	8000
0xFFFF	0x7FF0	0x0000 800F	4007	7FFF
0x8000		0x0000 0010	8	0010
0x0000		0xFFFF 8010	C008	8010

When oversampling mode is active, the SSATE and RSHIFT1..4 bits are not supported.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies of a buffer overrun event when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (eight stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_CR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun condition occurs, the ADC is still operating and can continue to convert unless the software decides to stop and reset the sequence by setting ADSTP to 1.

OVR flag is cleared by software by writing 1 to it.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the OVRMOD control bit of the ADC_CFGR register:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (8 data) and the new conversion is discarded and lost. If OVR remains at 1, any further conversion is be performed but the resulting data is also dicarded.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data.

Figure 210. Example of overrun (OVRMOD = 0)

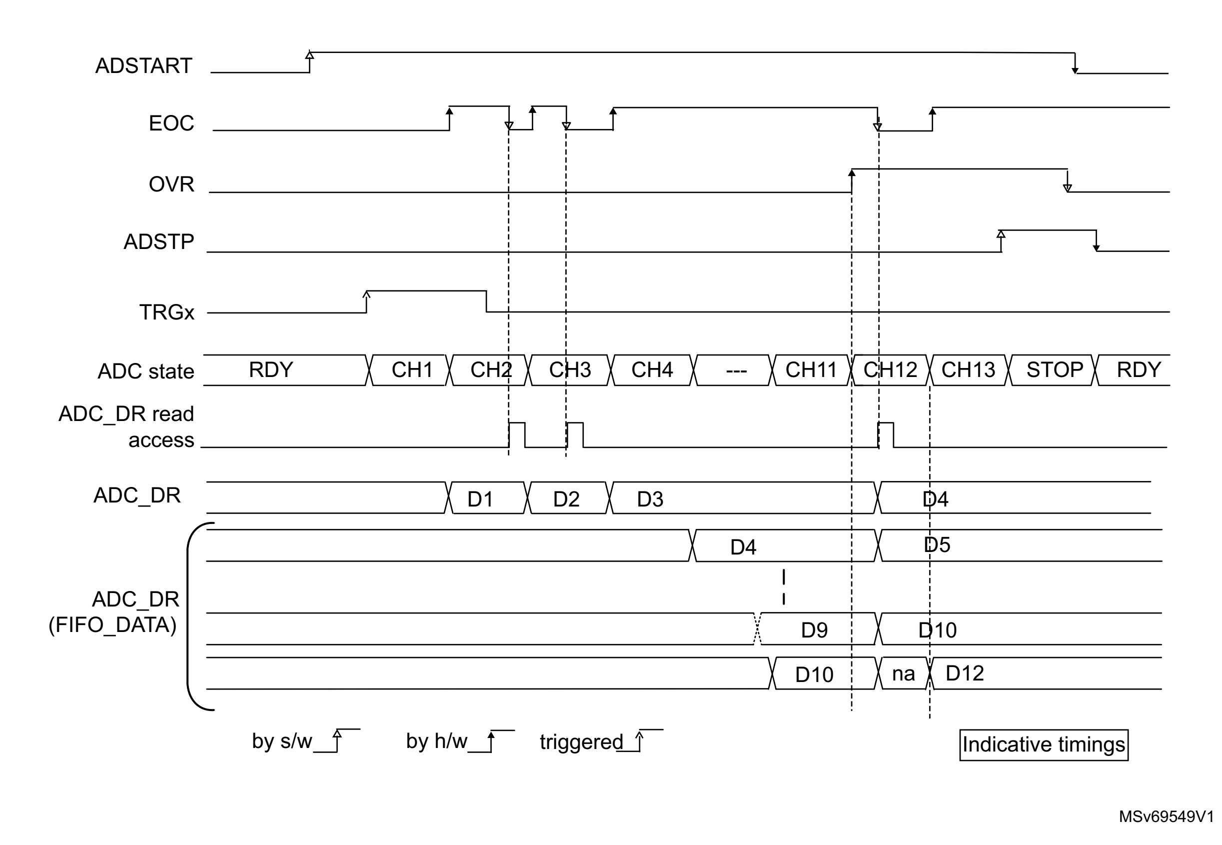
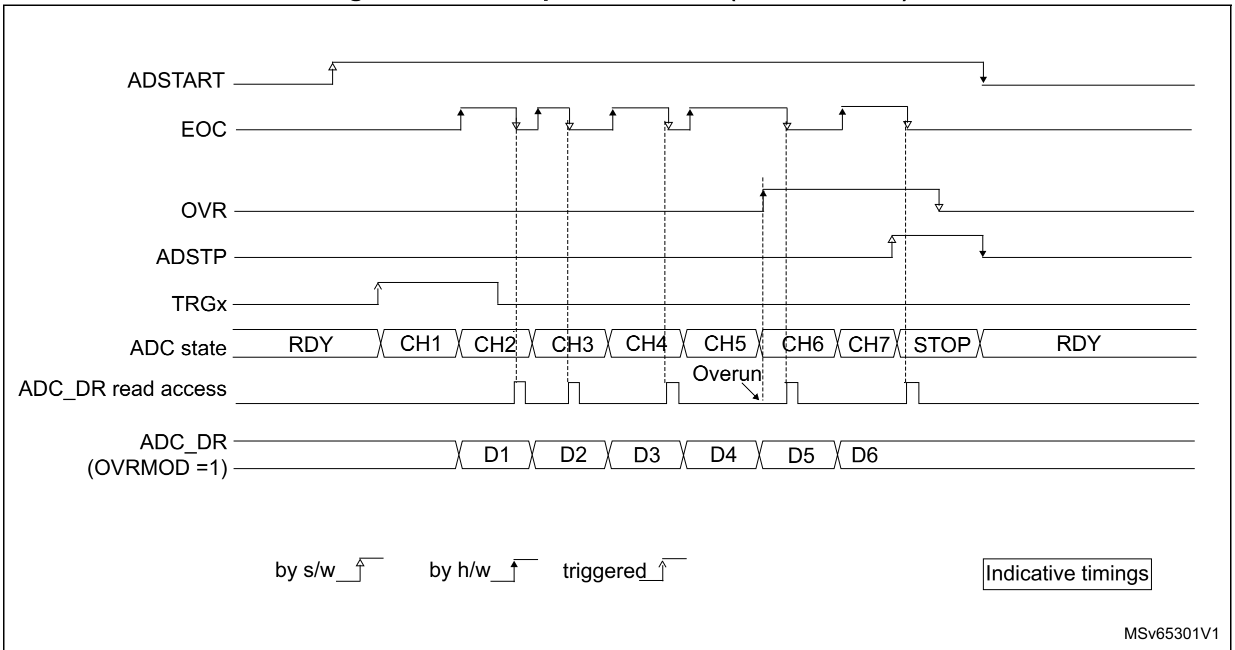


Figure 211. Example of overrun (OVRMOD = 1)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversion without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD should be configured to 0 to manage overrun events or FIFO overflows as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert, and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMNGT bit = 01 or 11 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR=1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMNGT of the ADC_CFGR register in single ADC mode, or with bit DAMDF of the ADC_CCR register in dual ADC mode:

- DMA one shot mode (DMNGT bit = 01).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT bit = 11)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMNGT=01)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMNGT=11)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

DMA with FIFO

The output data register has eight-stage FIFO. Two different DMA requests are generated parallel. When a data is available, “SREQ single request” generated, when 4 data are available, “BREQ burst request” generated. DMA2 can be programmed either single transfer mode or incremental burst mode(4 beats), according to this mode, correct request line is selected by the DMA2. Please refer to the DMA2 chapter for further information.

28.4.28 Managing conversions using the DFSDM

The ADC conversion results can be transferred directly to the Digital Filter for Sigma Delta Modulators (DFSDM).

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers 16 least significant bits of the regular data register data to the DFSDM, which in turns resets the EOC flag once the transfer is effective.

The data format must be 16-bit signed:

ADC_DR[31:16] = don’t care

ADC_DR[15] = sign

ADC_DR[14:0] = data

Any value above 16-bit signed format is truncated.

28.4.29 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY=1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 212*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 213*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN=0 or 1) and after each sequence of injected conversions (whatever JDISCEN=0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to re-start a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 213*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 215*).

The behavior is slightly different in auto-injected mode (JAUTO=1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 216*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO=1, CONT=1 and AUTDLY=1), follow the following procedure:

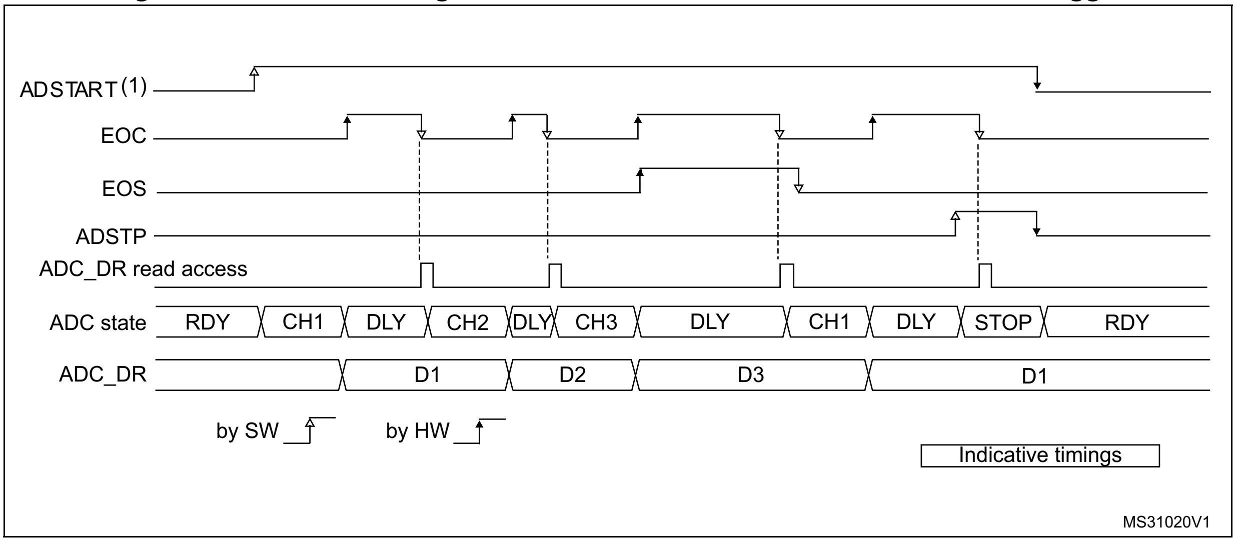
1. Wait until JEOS=1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP=1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can re-start if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

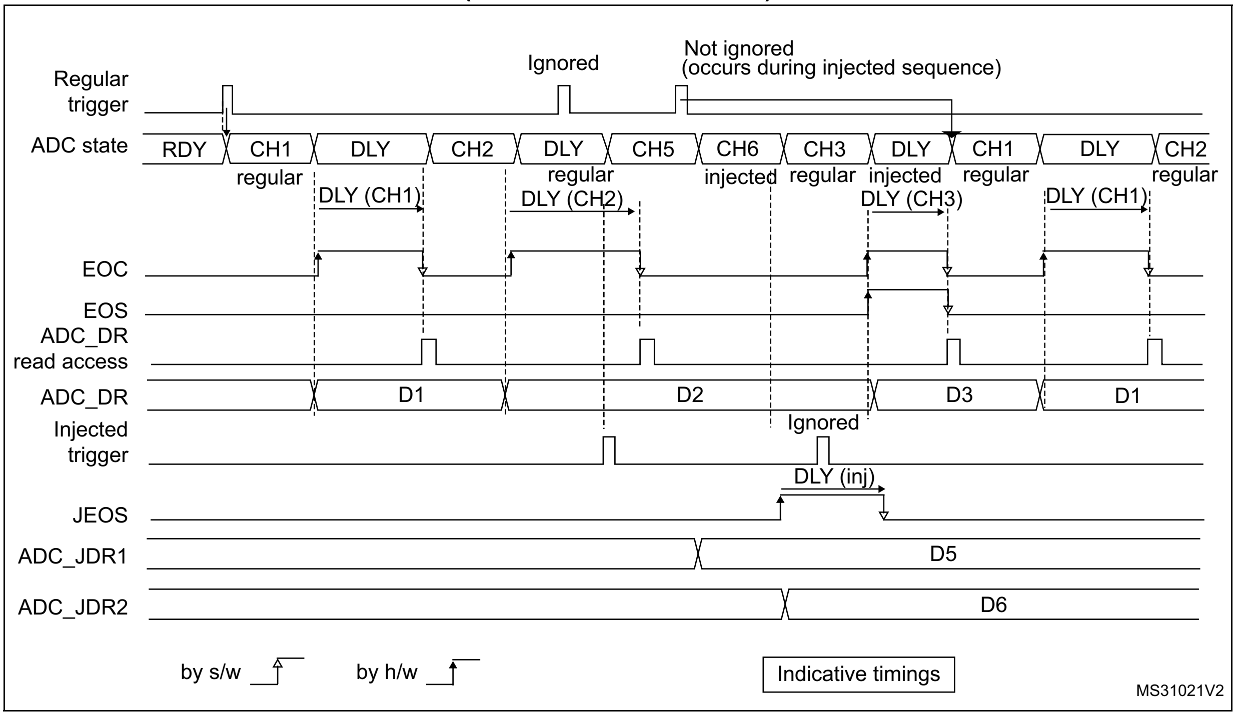
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 212. AUTDLY=1, regular conversion in continuous mode, software trigger



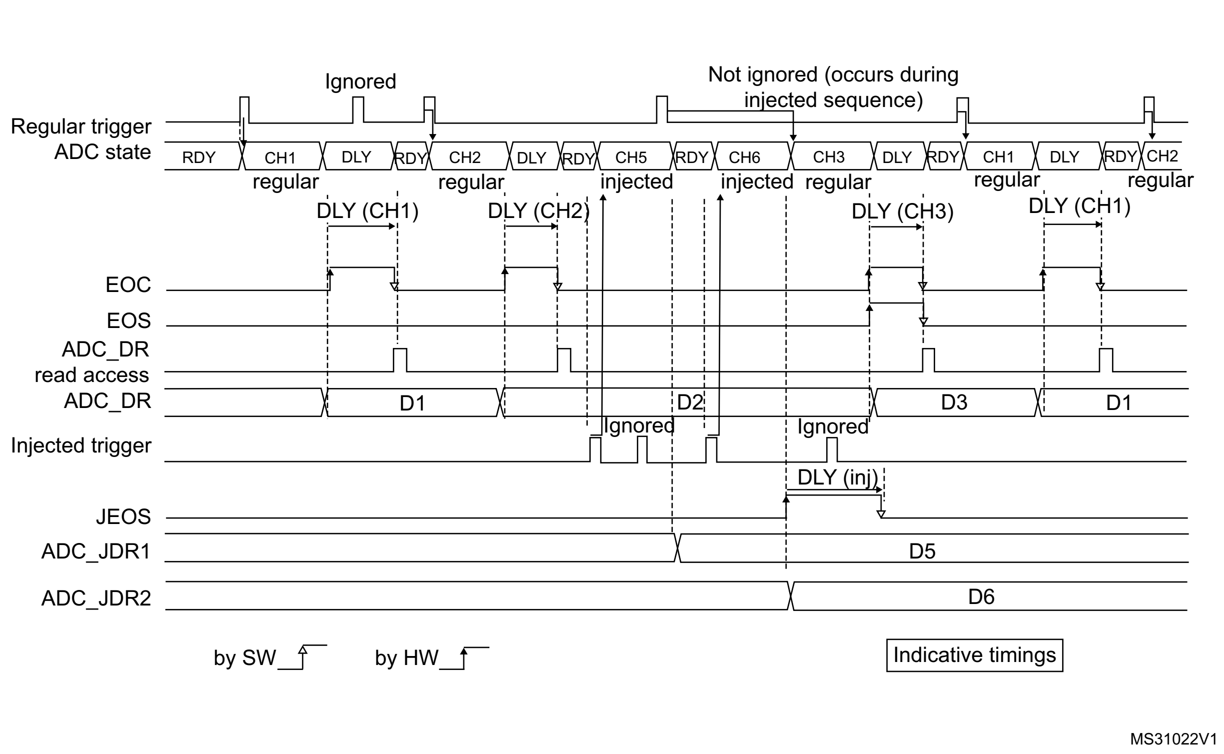
1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 213. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=0; JDISCEN=0)



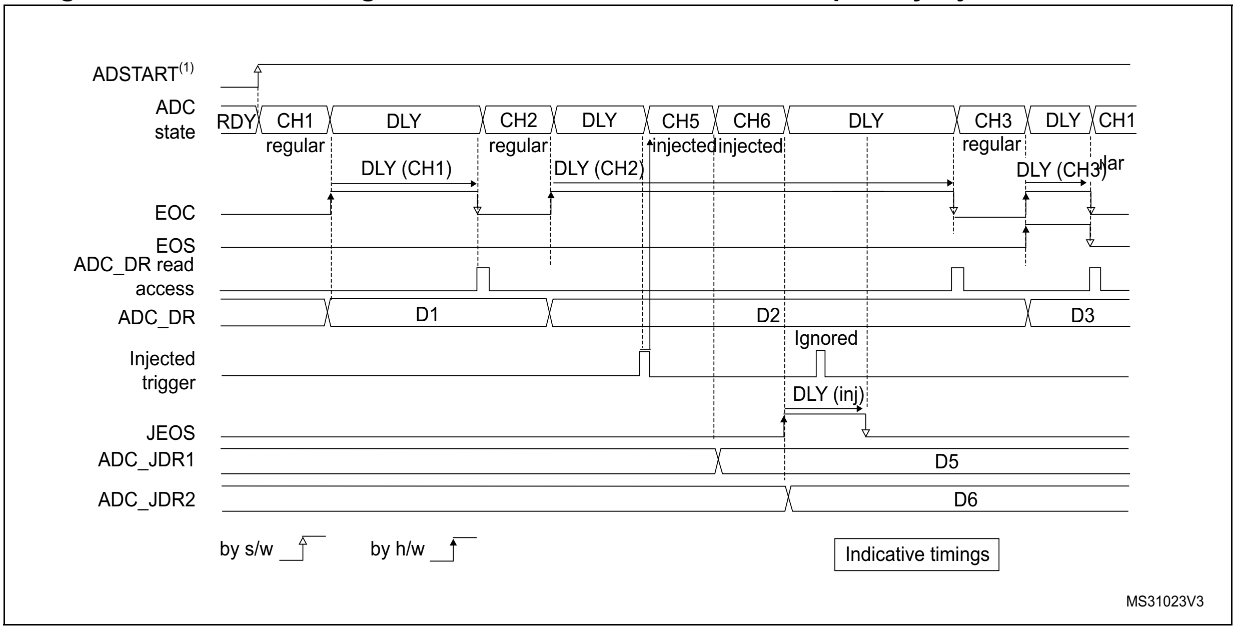
1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 214. AUTDLY=1, regular HW conversions interrupted by injected conversions (DISCEN=1, JDISCEN=1)



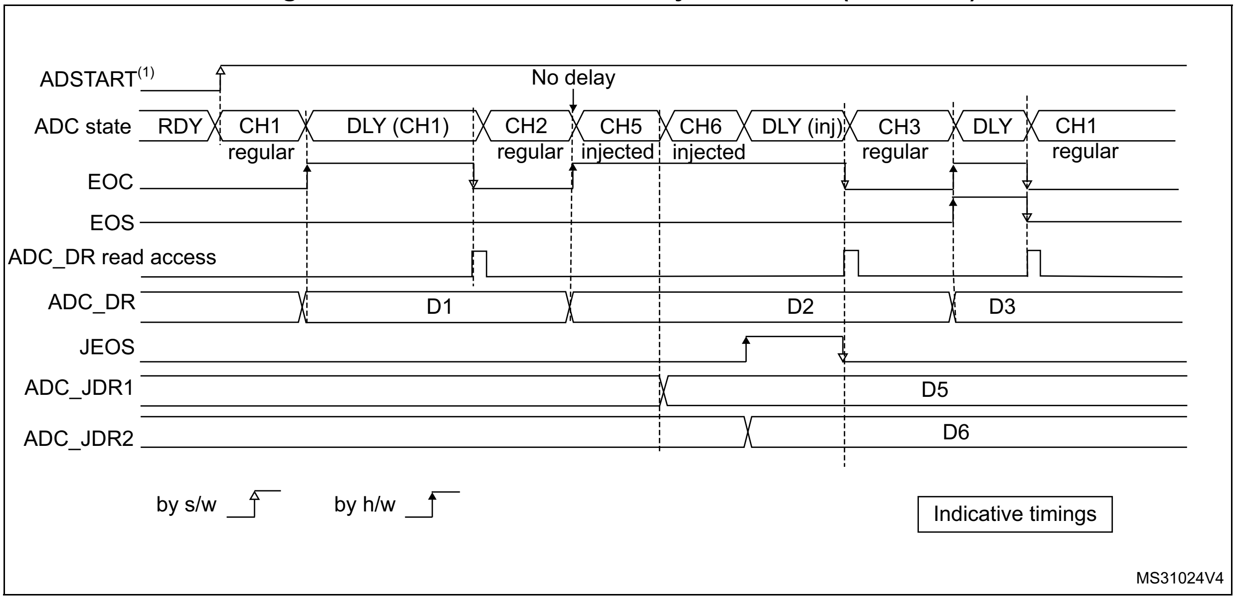
1. AUTDLY=1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT=0, DISCEN=1, DISCNUM=1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=1, CHANNELS = 5,6

Figure 215. AUTDLY=1, regular continuous conversions interrupted by injected conversions



1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN=0x1 (HW Trigger), JDISCEN=0, CHANNELS = 5,6

Figure 216. AUTDLY=1 in auto- injected mode (JAUTO=1)

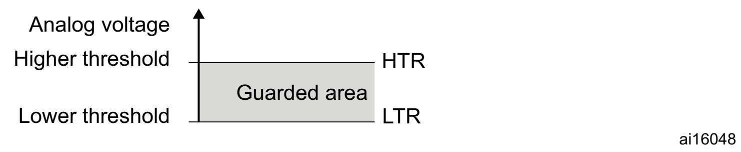


1. AUTDLY=1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT=1, DISCEN=0, CHANNELS = 1, 2
3. Injected configuration: JAUTO=1, CHANNELS = 5,6

28.4.30 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 217. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDyIE in the ADC_IER register (x=1,2,3).

AWDy (y=1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels[(1)] remain within a configured voltage range (window).

Table 196 shows how the ADC_CFGRy registers should be configured to enable the analog watchdog on one or more channels.

Table 196. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single[(1)] injected channel	1	0	1
Single[(1)] regular channel	1	1	0
Single[(1)] regular or injected channel	1	1	1

1. Selected by the AWDyCH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HTR1[25:0] of the ADC_HTR1 register and LTR1[25:0] of the ADC_LTR1 register for the analog watchdog 1.

The threshold can be up to 26-bits (16-bit resolution with oversampling, OSVR[9:0]=1024).

When converting data with a resolution of less than 16 bits (according to bits RES[2:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 16-bit converted data (left aligned to the half-word boundary).

Table 197 describes how the comparison is performed for all the possible resolutions for analog watchdog 1,2,3.

Table 197. Analog watchdog 1,2,3 comparison

Resolution (bit RES[2:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
16 bits	DATA[15:0]	LTR1[25:0] and HTR1[25:0]	-
14 bits	DATA[15:2],00	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[1:0] and HTR1[1:0] to 00
12 bits	DATA[15:4],0000	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[3:0] and HTR1[3:0] to 0000
10 bits	DATA[15:6],000000	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[5:0] and HTR1[5:0] to 000000
8 bits	DATA[15:8],00000000	LTR1[25:0] and HTR1[25:0]	User must configure LTR1[7:0] and HTR1[7:0] to 00000000

1. The watchdog comparison is performed on the raw converted data before any alignment calculation and before applying any offsets (the data which is compared is not signed).

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCHy[19:0] (y=2,3).

The corresponding watchdog is enabled when any bit of AWDCHy[19:0] (y=2,3) is set.

The threshold can be up to 26-bits (16-bit resolution with oversampling, OSVR[9:0]=1024) and are programmed with the ADC_HTR2, ADC_LTR2, ADC_LTR3, and ADC_HTR3 registers.

When converting data with a resolution of less than 16 bits (according to bits RES[2:0]), the LSBs of the programmed thresholds must be kept cleared, the internal comparison being performed on the full 16-bit converted data (left aligned to the half-word boundary).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCx_AWDy_OUT (x=ADC number, y=watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCx_AWDy_OUT signal as ETR.

ADCx_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADCx_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCx_AWDy_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCx_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS=1). Note that stopping regular or injected conversions (setting ADSTP=1 or JADSTP=1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDy flag has no influence on the generation of ADCx_AWDy_OUT (ex: ADCy_AWDy_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 218. ADCy_AWDx_OUT signal generation (on all regular channels)

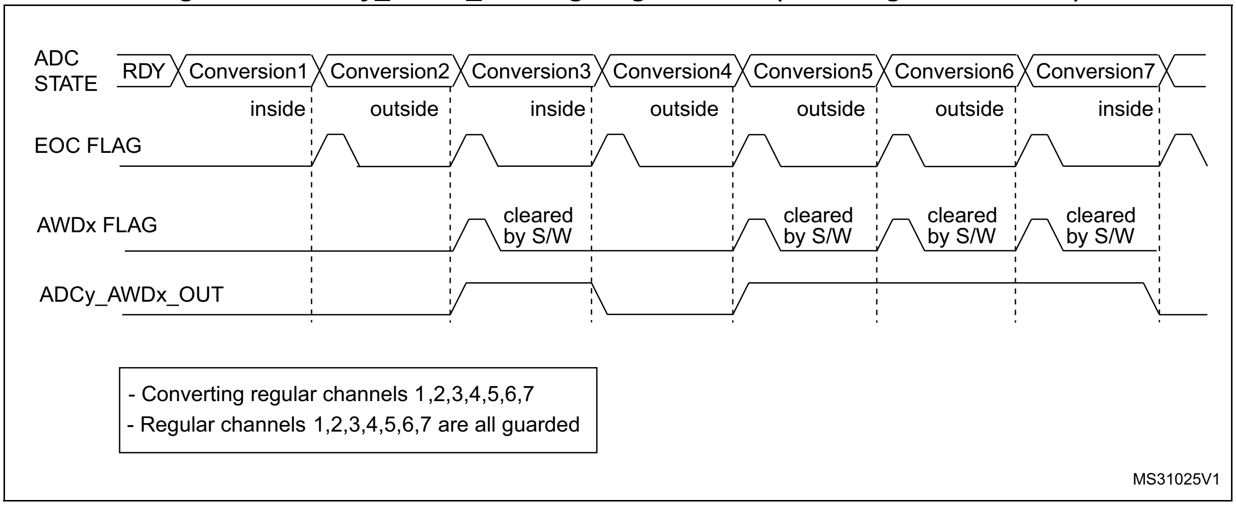


Figure 219. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by SW)

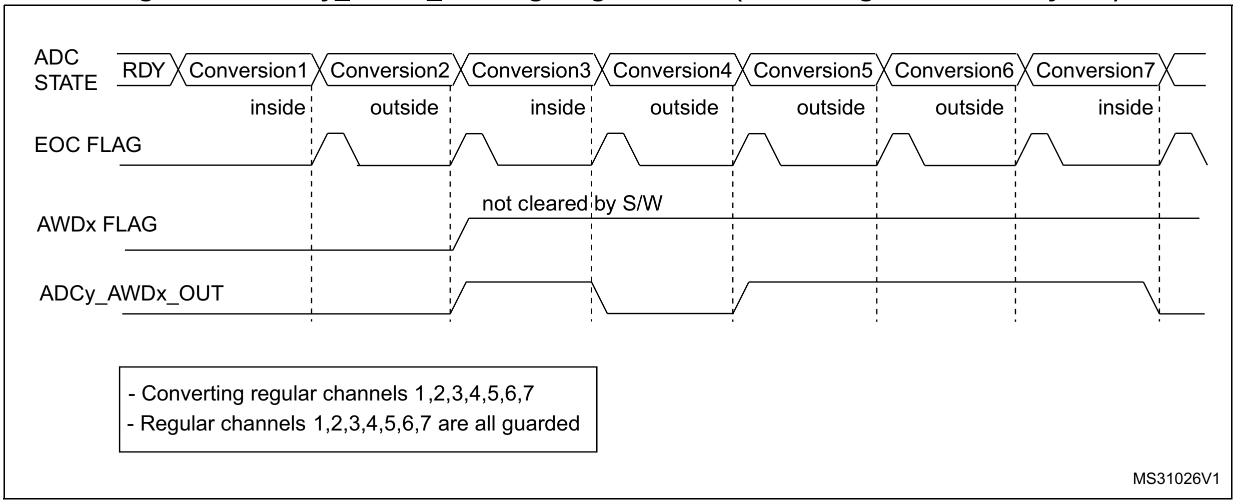


Figure 220. ADCy_AWDx_OUT signal generation (on a single regular channel)

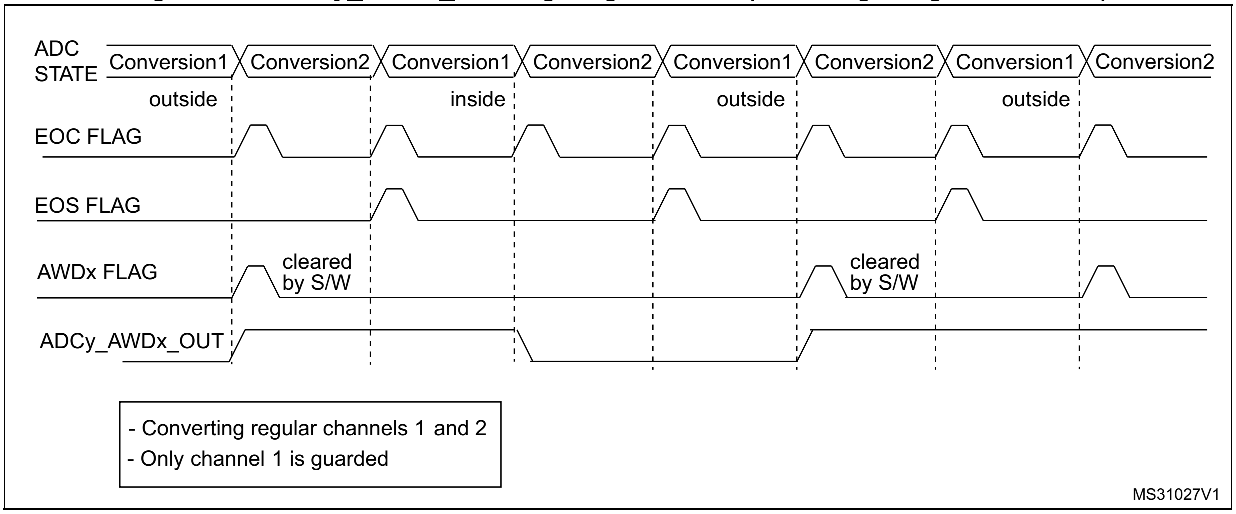
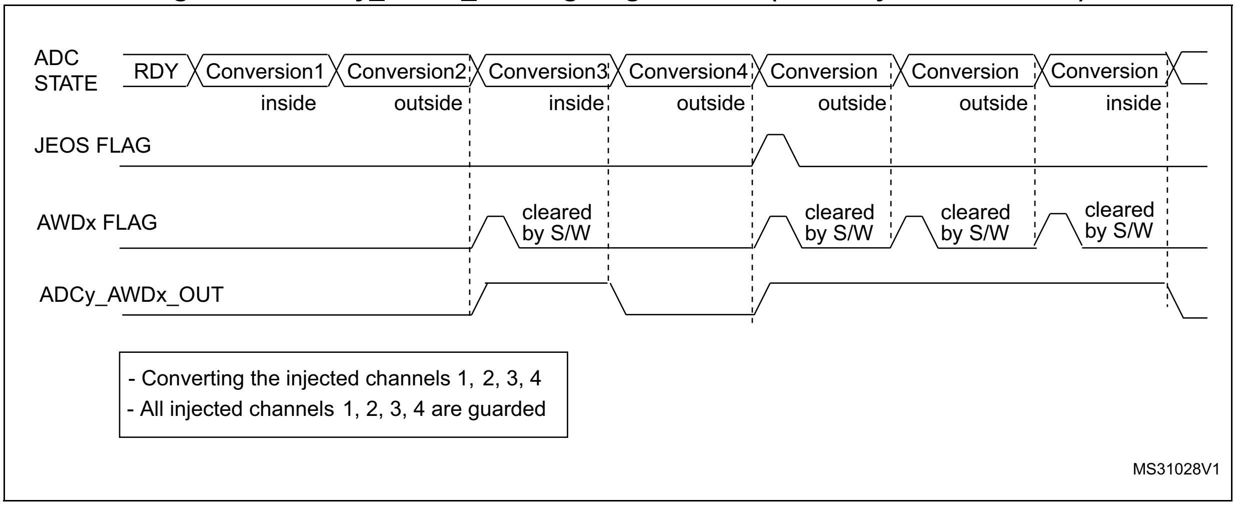


Figure 221. ADCy_AWDx_OUT signal generation (on all injected channels)



28.4.31 Oversampler

The oversampling unit performs data preprocessing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 26-bit (16-bit values and OSVR[9:0] = 1024).

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

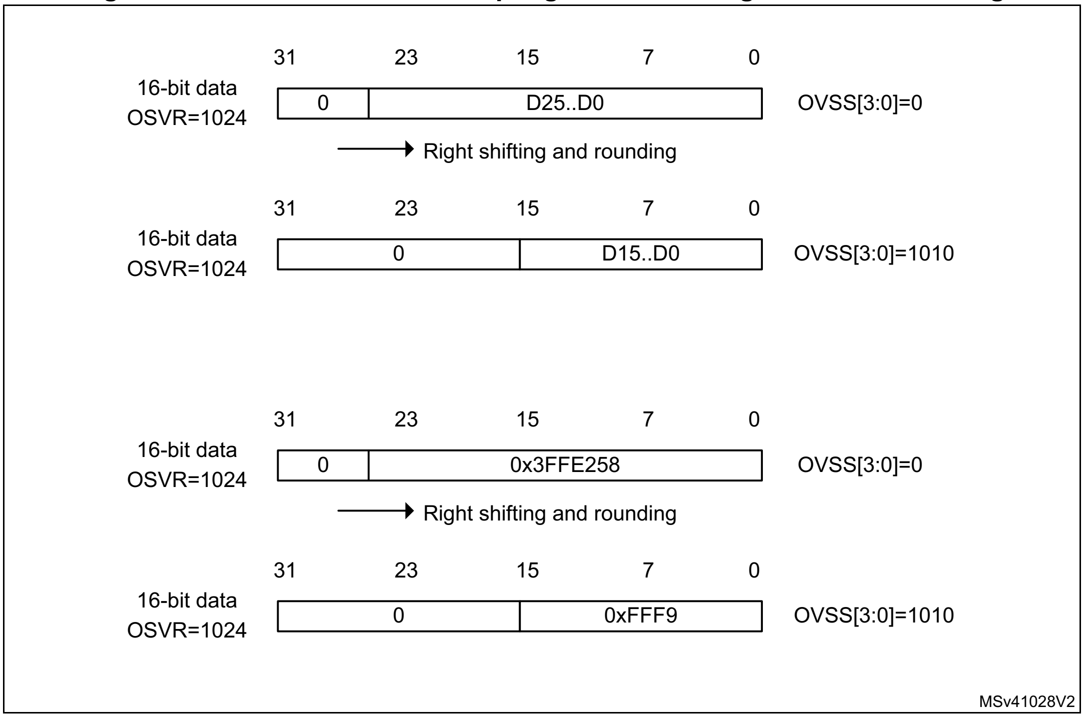
It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OSVR[9:0] bits in the ADC_CFGR2 register, and can range from 2x to 1024x. The division coefficient M consists of a right bit shift up to 10 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 26 bits (1024 x 16-bit results), which can be left or right shifted. When right shifting is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being transferred into the ADC_DR data register.

The *Table 222* gives a numerical example of the processing, from a raw 26-bit accumulated data to the final 16-bit result.

Figure 222. 16-bit result oversampling with 10-bits right shift and rouding



There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follow:

- the end of the sampling phase (EOSMP) is set after each sampling phase
- the end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- the end of sequence (EOS) occurs once the sequence of oversampled data is completed (i.e. after N x sequence length conversions total)

Single ADC operating modes support when oversampling

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous mode conversions
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[2:0] bits in ADC_CFGR register) are accumulated, truncated, rounded and shifted in the same way as 16-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The data are always provided right-aligned.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSETy_EN bit in ADC_OFRy register is ignored (considered as reset).

Analog watchdog

The analog watchdog functionality is maintained (AWDSGL and AWDEN bits), with the following difference:

- the RES[2:0] bits are ignored, comparison is always done on using the full 26-bit values HTRx[25:0] and LTRx[25:0]
- *the comparison is performed on the oversampled accumulated value before shifting*

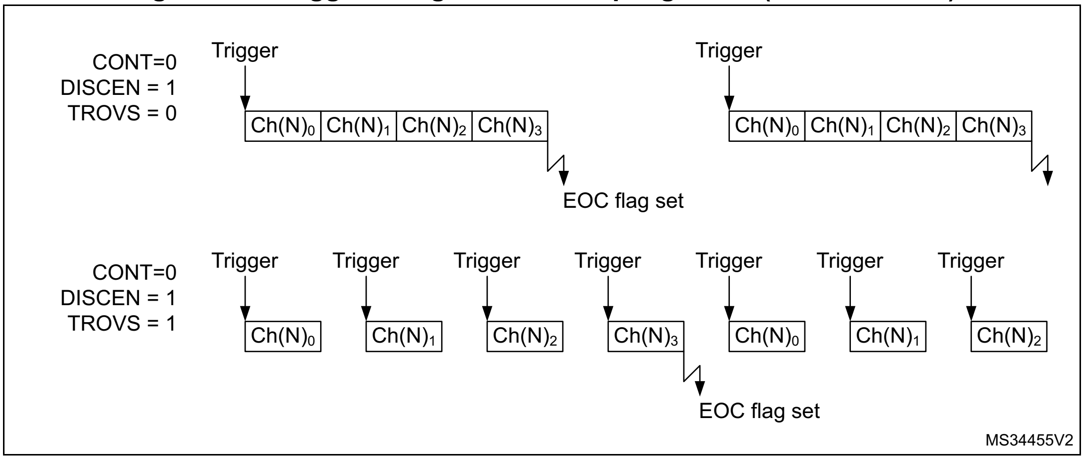
Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The *Figure 223* below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 223. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

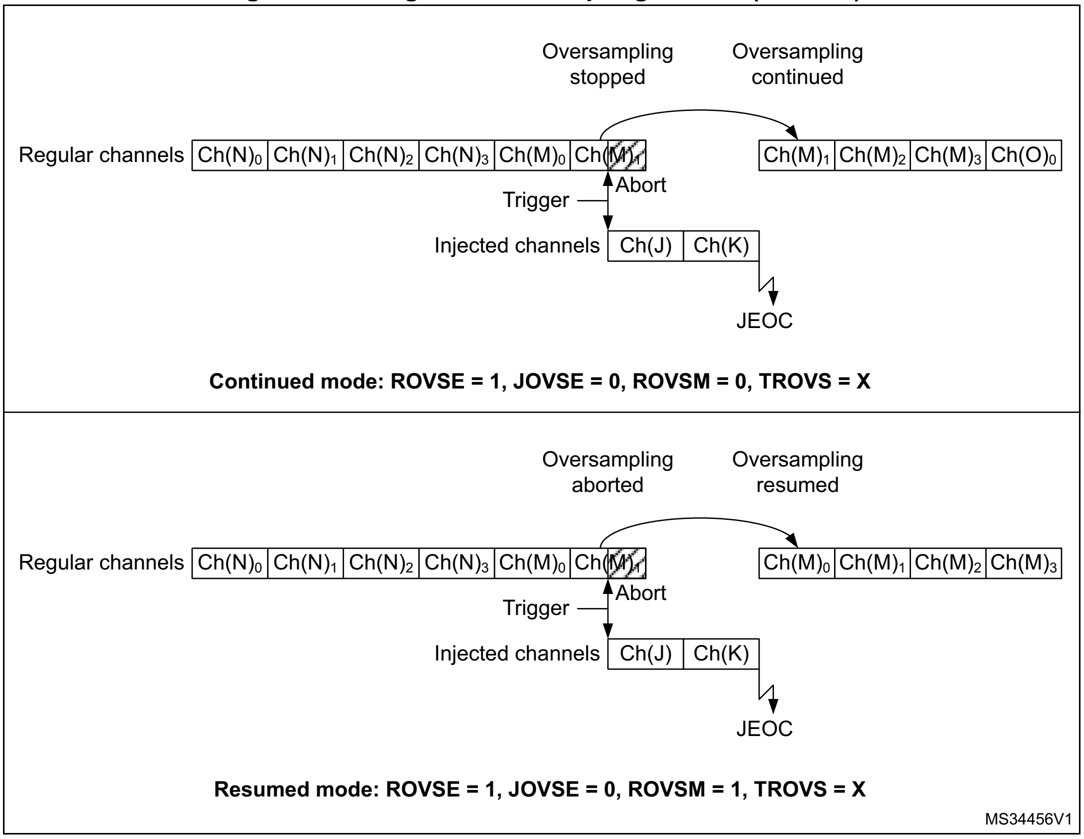
Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- in continued mode, the accumulation re-starts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- in resumed mode, the accumulation re-starts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequencer is blocked.

The *Figure 224* gives examples for a 4x oversampling ratio.

Figure 224. Regular oversampling modes (4x ratio)



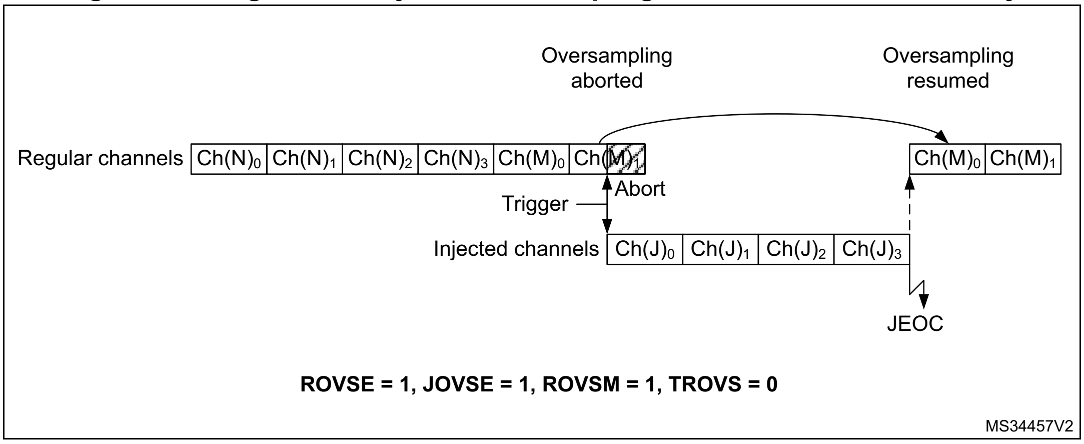
Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on *Figure 225* below.

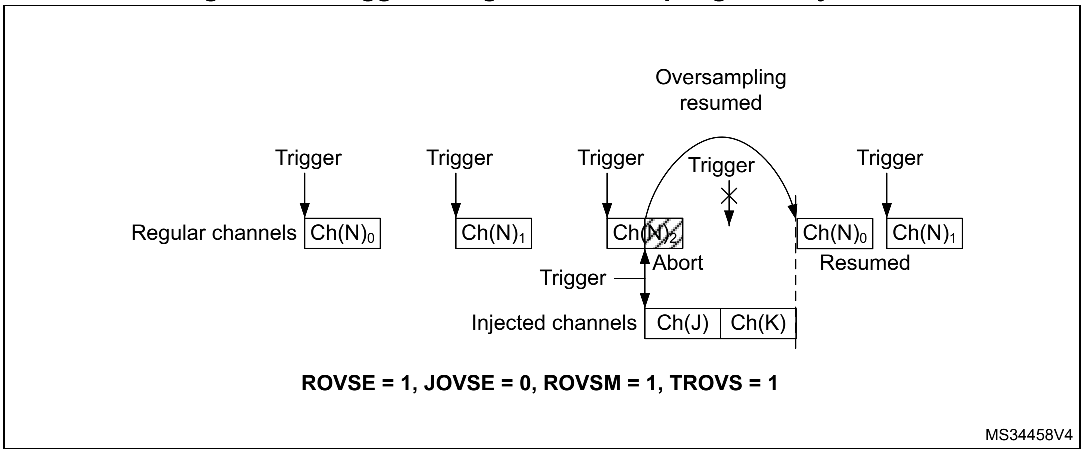
Figure 225. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on *Figure 226* below.

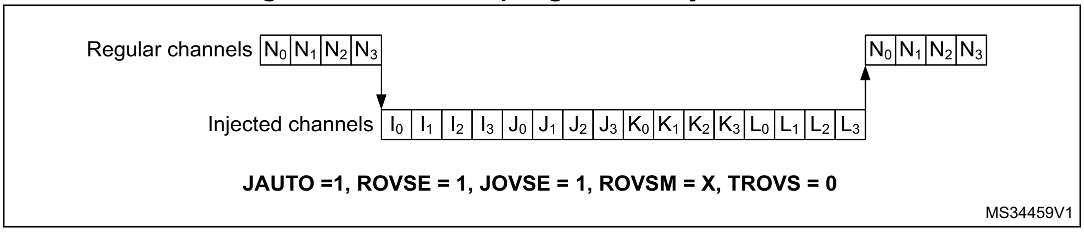
Figure 226. Triggered regular oversampling with injection



Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 227* below shows how the conversions are sequenced.

Figure 227. Oversampling in auto-injected mode



It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO=1, DISCEN=0, JDISCEN=0, ROVSE=1, JOVSE=1 and TROVSE=1.

Dual ADC modes support when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The *Table 198* below summarizes all combinations, including modes not supported.

Table 198. Oversampler operating modes summary

Regular Over-sampling ROVSE	Injected Over-sampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered Regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

28.4.32 Dual ADC modes

In devices with two ADCs or more, dual ADC modes can be used (see *Figure 228*):

- ADC1 and ADC2 can be used together in dual mode (ADC1 is master)

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADC_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

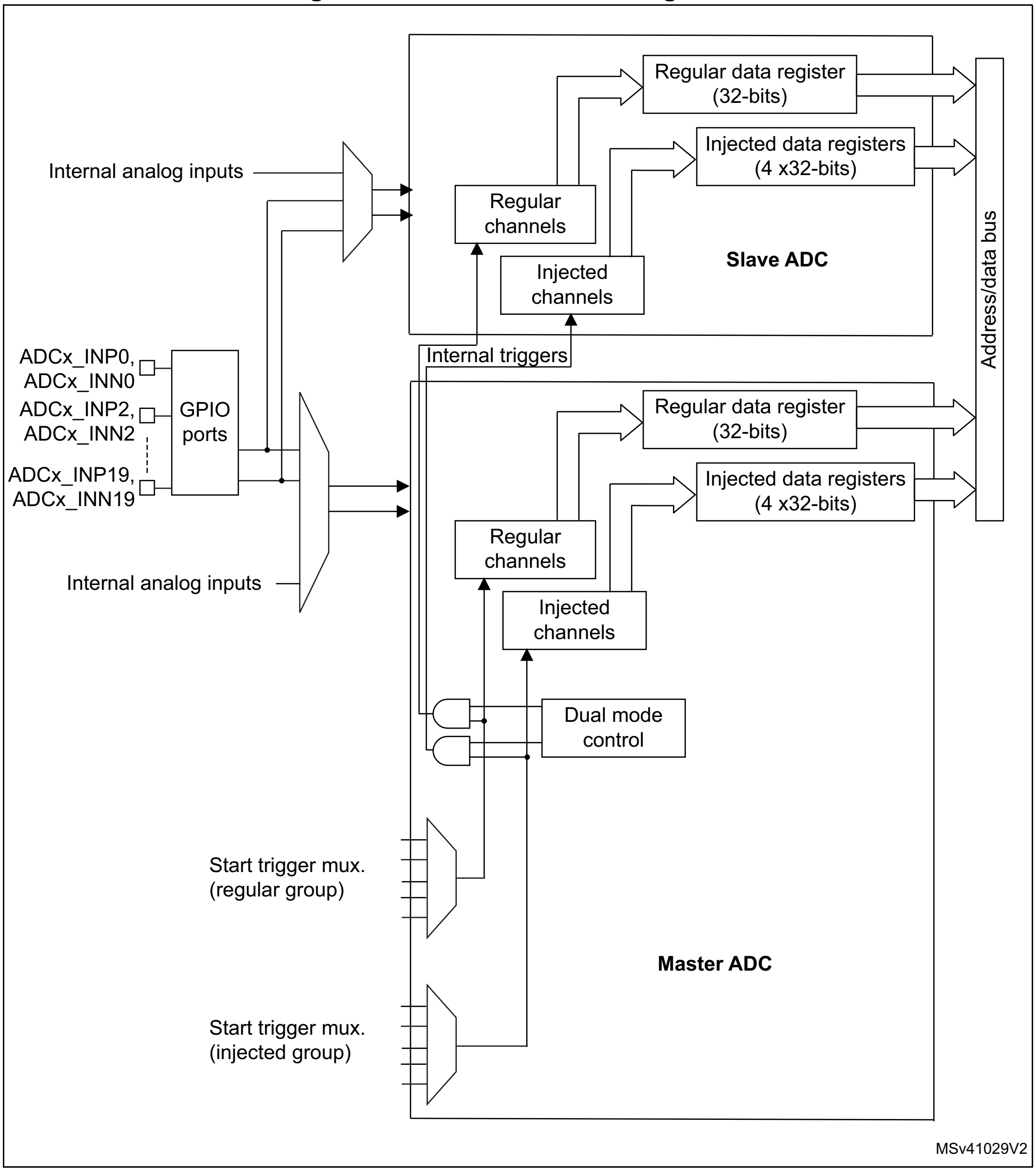
To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADC_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADC_CSR).

Figure 228. Dual ADC block diagram(1)



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0]=00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL[4:0] bits in the ADC_JSQR register).

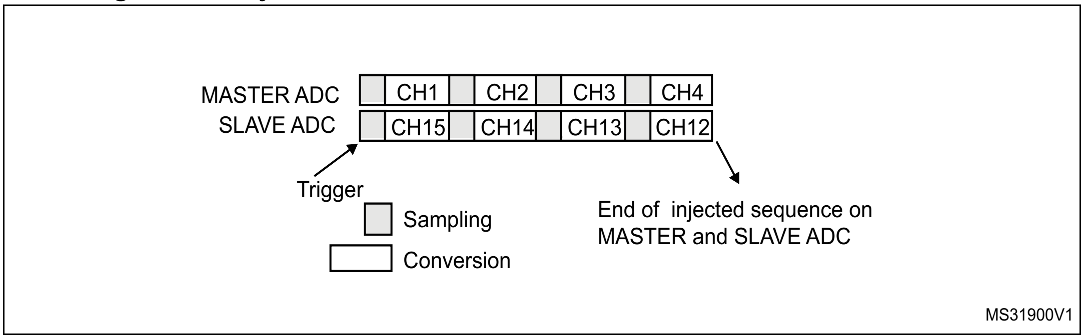
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length and inside a sequence, the N-th conversion in master ans slave must be configured with the same sampling time.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in *Figure 229*), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 229. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN=1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL[4:0] bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) aborts the current simultaneous conversions, which are re-started once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length and inside a sequence, the N-th conversion in master ans slave must be configured with the same sampling time.

Software is notified by interrupts when it can read the data:

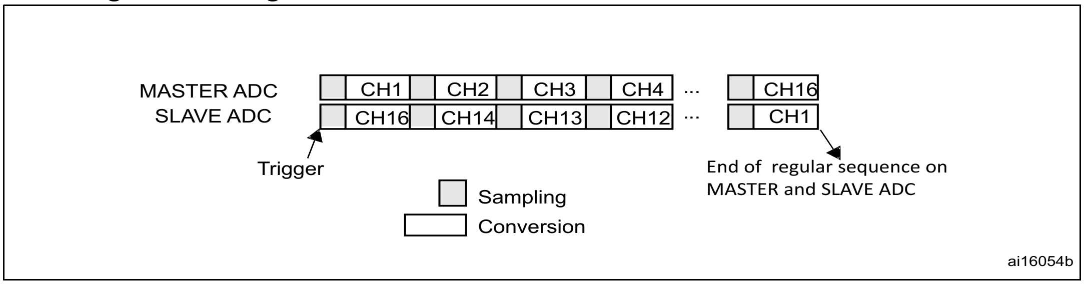
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in *Figure 230*), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADC_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits DAMDF[1:0] must be kept cleared.
  – Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  – Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configuring dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  – Configure DAMDF[1:0]=0b10 or 0b11 (depending on resolution).
  – A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR)
  – A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CDR register.
  – both EOC flags are cleared when the DMA reads the ADC_CDR register.

Note: When DAMDF[1:0]=0b10 or 0b11, the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 230. Regular simultaneous mode on 16 channels: dual ADC mode



If DISCEN=1 then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur (“n” is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADC_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADC_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multiple-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates 2 conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. This delay starts to count after the end of the sampling phase of the master conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADC_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure DAMDF[1:0]=0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CCR register.
- Both EOC flags are cleared when the DMA reads the ADC_CCR register.

Figure 231. Interleaved mode on 1 channel in continuous conversion mode: dual ADC mode

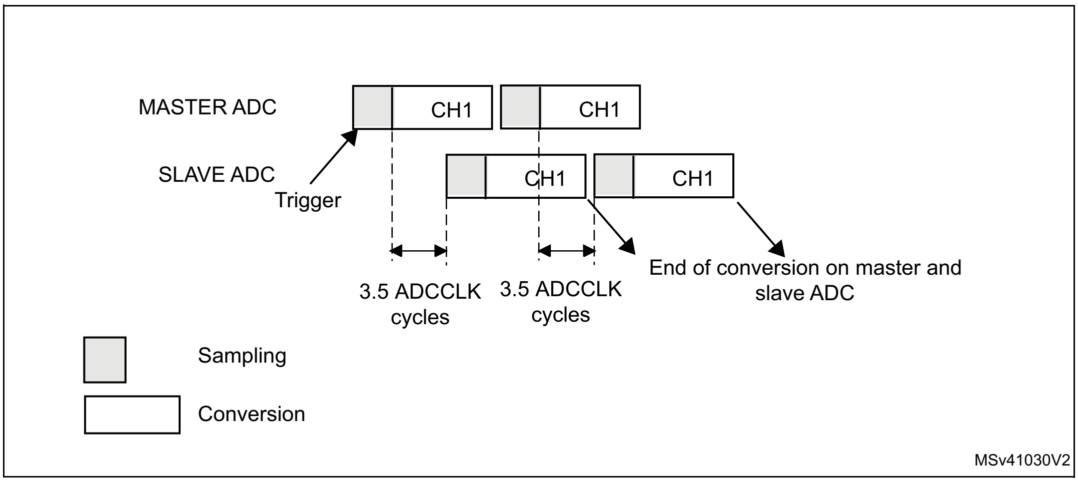
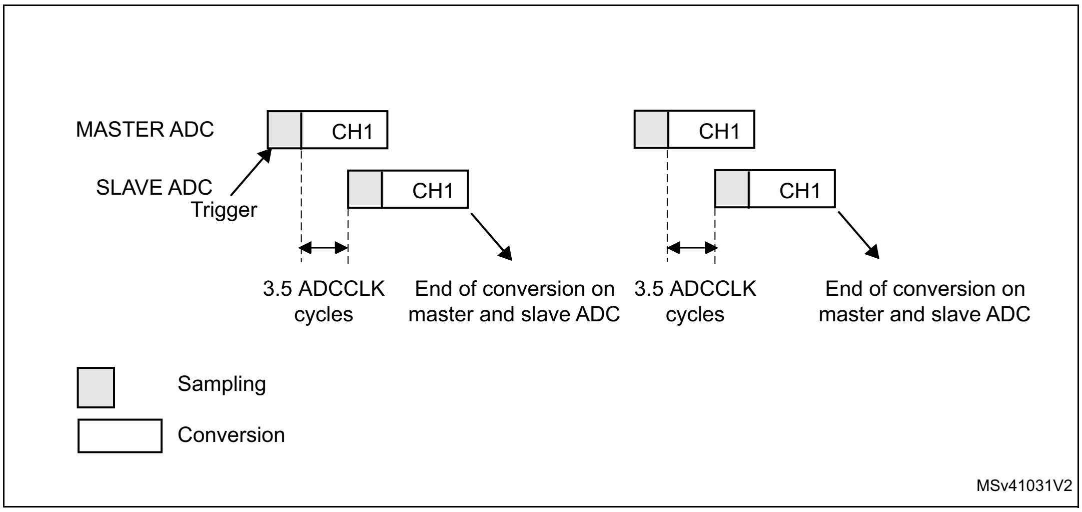


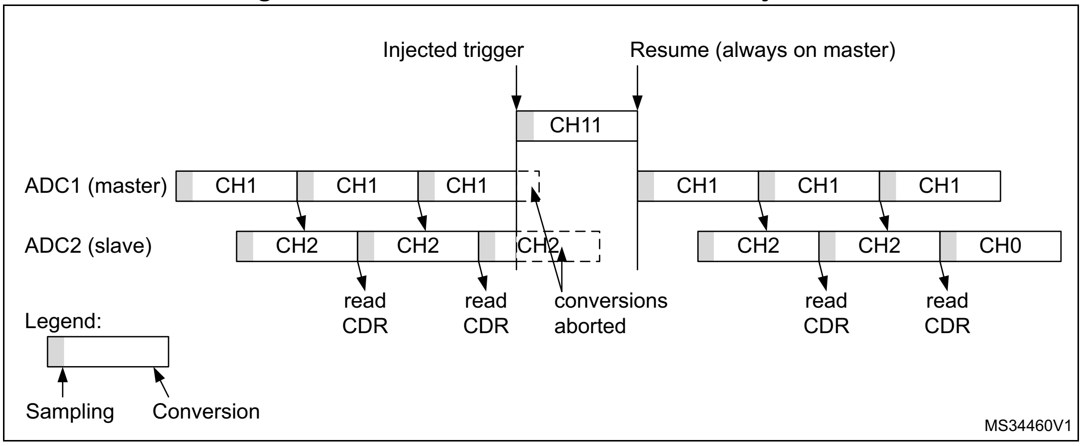
Figure 232. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



If DISCEN=1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is re-started from the master (see *Figure 233* below).

Figure 233. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.

Injected discontinuous mode disabled (JDISCEN=0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

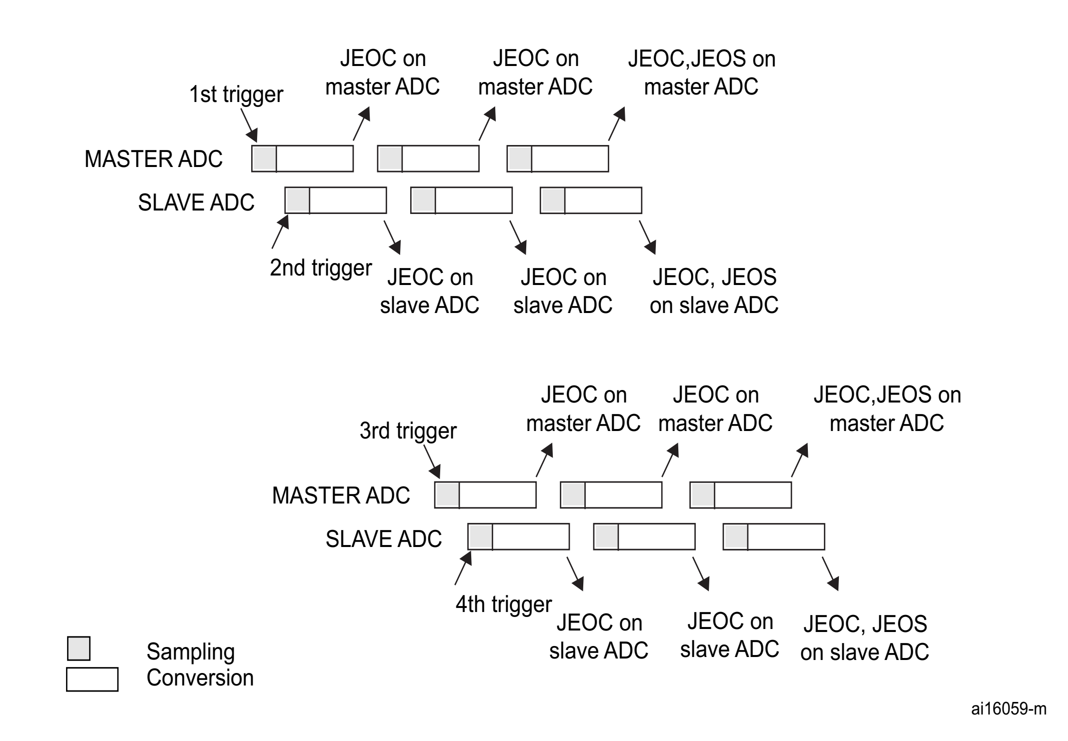
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 234. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN=1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

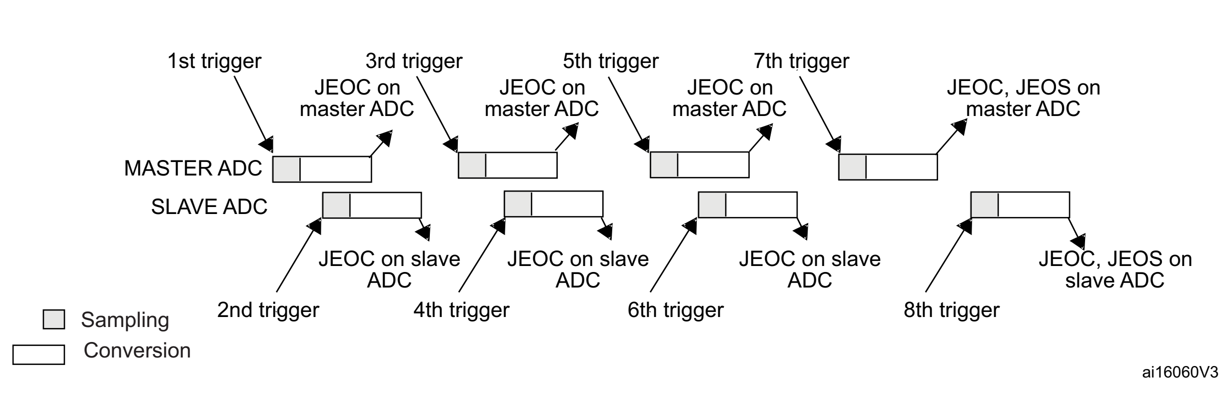
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 235. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: The sequences must be converted with the same length, the N-th conversion in master and slave mode must be configured with the same sampling time inside a given sequence, or the interval between triggers has to be longer than the long conversion time of the 2 sequences. If the above conditions are not respected, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

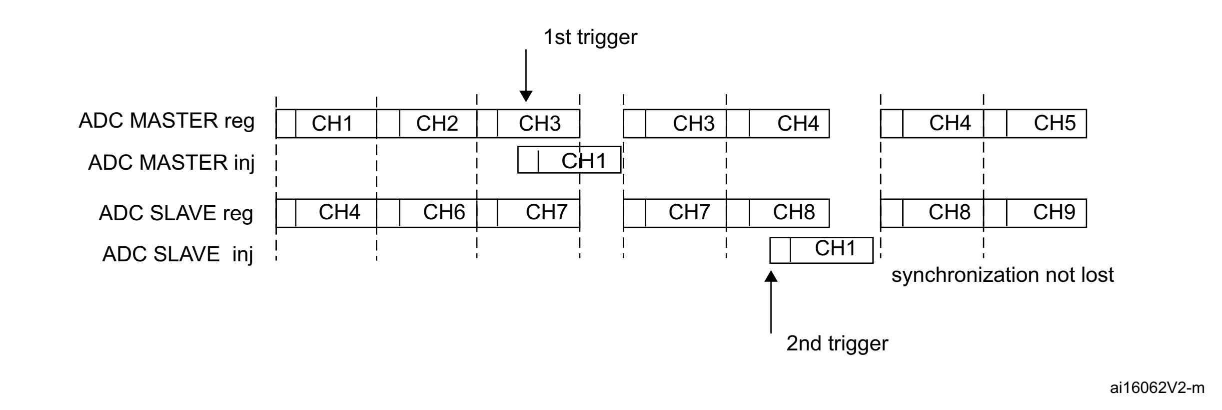
This mode is selected by programming bits DUAL[4:0]=00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 236* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

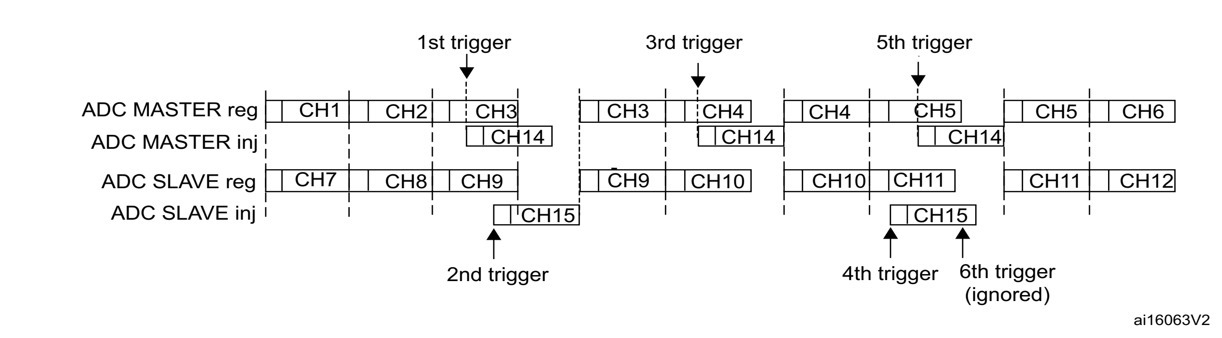
Note: The sequences must be converted with the same length, the N-th conversion in master and slave mode must be configured with the same sampling time inside a given sequence, or the interval between triggers has to be longer than the long conversion time of the 2 sequences. If the above conditions are not respected, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 236. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 237* shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 237. Case of trigger occurring during injected conversion



Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0]=00011.

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is alway the master's one. *Figure 238*, *Figure 239* and *Figure 240* show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.

Figure 238. Interleaved single channel CH0 with injected sequence CH11, CH12

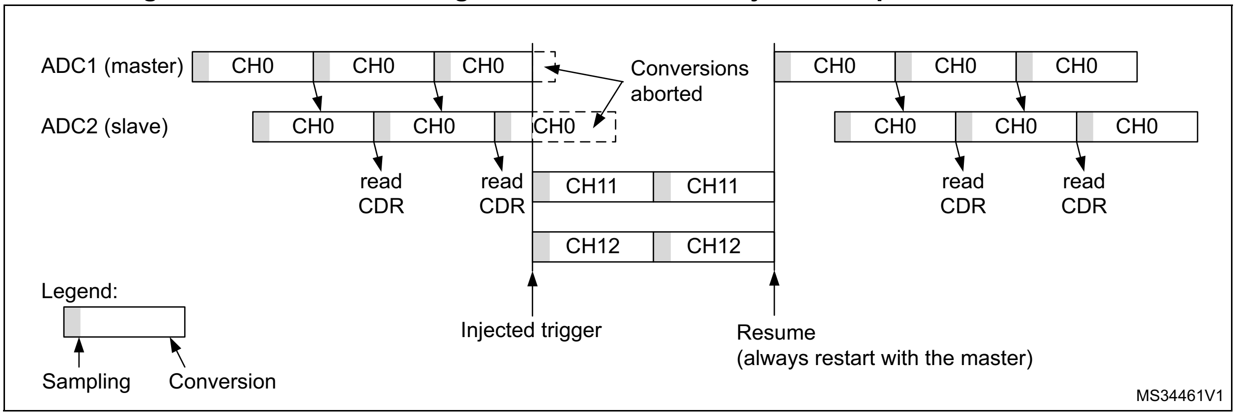


Figure 239. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

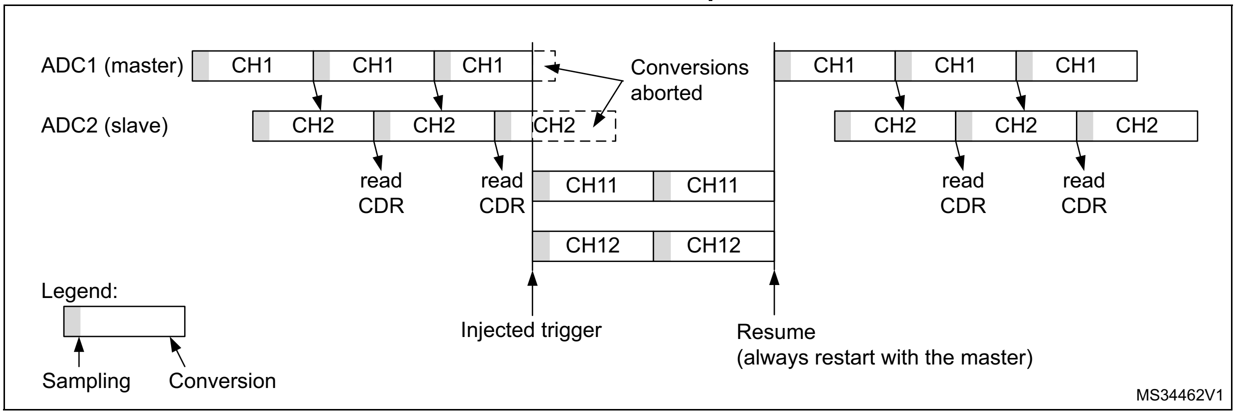
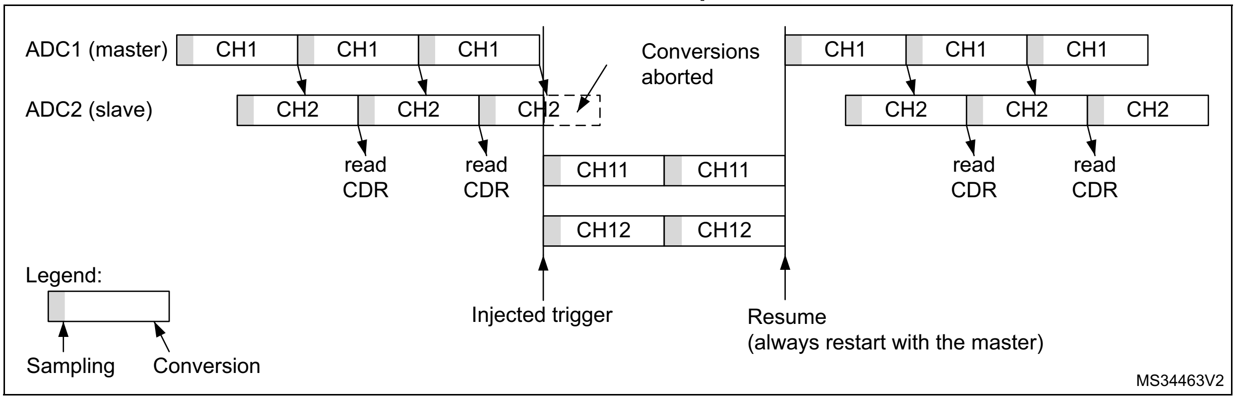


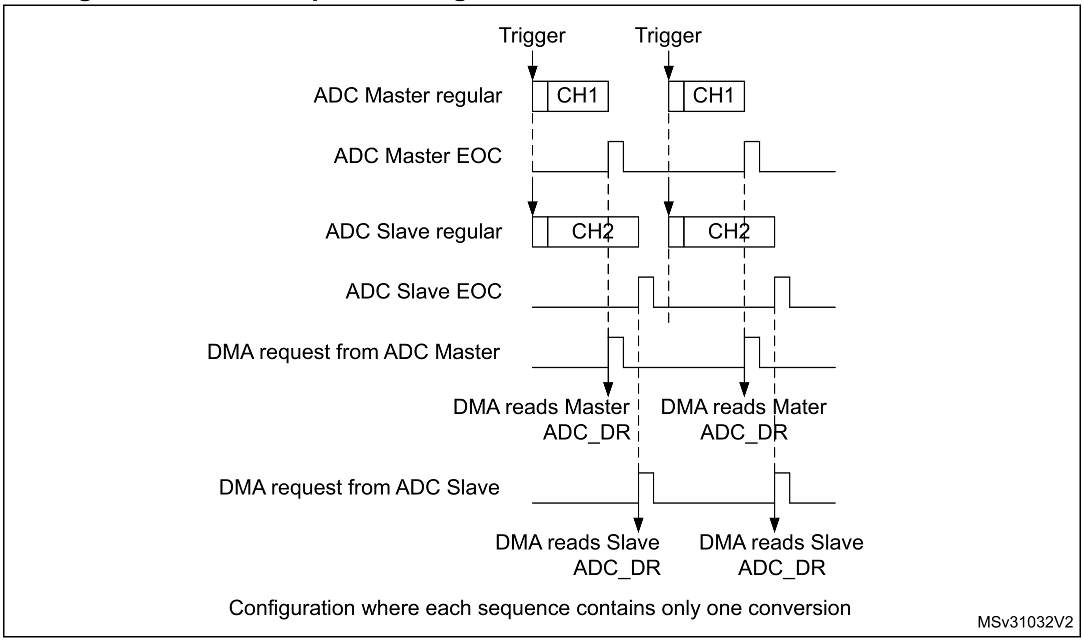
Figure 240. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 241: DMA Requests in regular simultaneous mode when DAMDF=0b00*).

Figure 241. DMA Requests in regular simultaneous mode when DAMDF=0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this DAMDF bits must be configured in the ADC_CCR register:

- **DAMDF=0b10, 32-bit format**: A single DMA request is generated alternatively when either the master or slave EOC events have occurred. At that time, the data items are alternatively available in the ADC_CDR2 32-bit register. This mode is used in interleaved mode and in regular simultaneous mode when resolution is above 16-bit.
  Example:

  Interleaved dual mode: a DMA request is generated each time a new 32-bit data is available:

  1st DMA request: ADC_CDR2[31:0] = MST_ADC_DR[31:0]

  2nd DMA request: ADC_CDR2[31:0] = SLV_ADC_DR[31:0]
- **DAMDF=0b10, 16-bit format**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADC_CDR contains the two half-words representing two ADC-

converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.

This mode is used in interleaved mode and in regular simultaneous mode when resolution is ranging from 10 to 16-bit. Any value above 16-bit in the master or the slave converter is truncated to the least 16 significant bits.

Example:

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: ADC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

2nd DMA request: ADC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 242. DMA requests in regular simultaneous mode when DAMDF=0b10

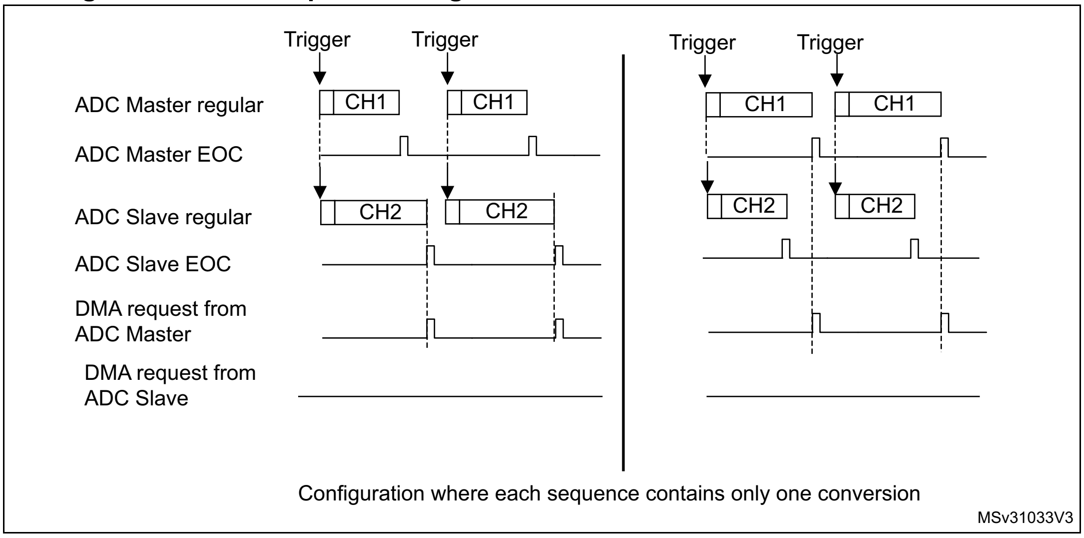
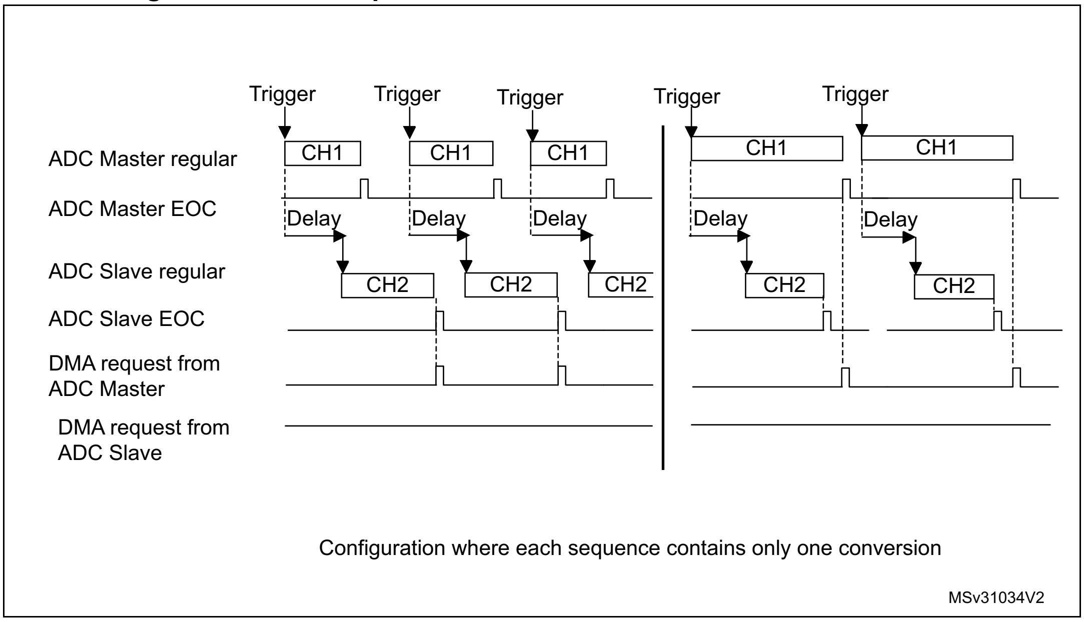


Figure 243. DMA requests in interleaved mode when DAMDF=0b10



Note: When using multiple-ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **DAMDF=0b11**: This mode is similar to the DAMDF=0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when the result is 8-bit. A new DMA request is issued when 4 new 8-bit values are available.

  Example:

  Interleaved dual mode: a DMA request is generated each time 4 data items are available (t0, t1,... are corresponding to the consecutive sampling instants)

  1st DMA request:

  ADC_CDR[7:0] = MST_ADC_DR[7:0]$_{t0}$

  ADC_CDR[15:8] = SLV_ADC_DR[7:0]$_{t0}$

  ADC_CDR[23:16] = MST_ADC_DR[7:0]$_{t1}$

  ADC_CDR[31:24] = SLV_ADC_DR[7:0]$_{t1}$

  2nd DMA request:

  ADC_CDR[7:0] = MST_ADC_DR[7:0]$_{t2}$

  ADC_CDR[15:8] = SLV_ADC_DR[7:0]$_{t2}$

  ADC_CDR[23:16] = MST_ADC_DR[7:0]$_{t3}$

  ADC_CDR[31:24] = SLV_ADC_DR[7:0]$_{t3}$

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when multiple-ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), bit DMNGT[1:0]=0b10 in the master ADC's ADC_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*.

Stopping the conversions in dual ADC modes

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

DFSDM mode in dual ADC mode interleaved mode

In dual ADC interleaved modes, the ADC conversion results can be transferred directly to the Digital Filter for Sigma Delta Modulators (DFSDM).

This mode is enabled by setting the bits DMNGT[1:0] = 0b10 in the master ADC's ADC_CFGR register.

The ADC transfers alternatively the 16 least significant bits of the regular data register from the master and the slave converter to a single channel of the DFSDM.

The data format must be 16-bit signed:

ADC_DR[31:16] = 0x0000

ADC_DR[15] = sign

ADC_DR[14:0] = data

Any value above 16-bit signed format in any converter is truncated.

DFSDM mode in dual ADC simultaneous mode

The dual mode is not required to use DFSDM in dual ADC simultaneous mode since conversion data are treated by each individual channel. Single mode with same trigger source results in simultaneous conversion with DFSDM interface.

28.4.33 Temperature sensor

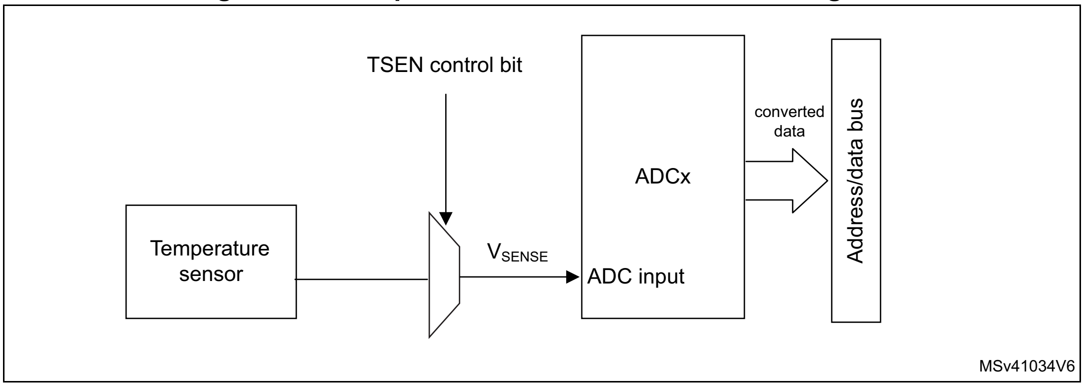
The temperature sensor can measure the junction temperature (T_J) of the device in the –40 to 125 °C temperature range.

The temperature sensor is internally connected to an ADC internal channel which is used to convert the sensor's output voltage to a digital value (refer to *Table ADC interconnection* in *Section 28.4.2: ADC pins and internal signals*). The sampling time for the temperature sensor's analog pin must be greater than the stabilization time specified in the product datasheet.

When not in use, the sensor can be put in power-down mode.

Figure 244 shows the block diagram of the temperature sensor.

Figure 244. Temperature sensor channel block diagram



Note: The TSEN bit must be set to enable the conversion of the corresponding ADC internal channel (temperature sensor, V_{SENSE}).

Reading the temperature

To use the sensor:

1. Select the ADC input channels (with the appropriate sampling time).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the TSEN bit in the ADCx_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS\_CAL2\_TEMP} - \text{TS\_CAL1\_TEMP}}{\text{TS\_CAL2} - \text{TS\_CAL1}} \times (\text{TS\_DATA} - \text{TS\_CAL1}) + \text{TS\_CAL1\_TEMP}$$

Where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP
- TS_DATA is the actual temperature sensor output value converted by ADC

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and SENSEEN bits should be set at the same time.

28.4.34 V_{BAT} supply monitoring

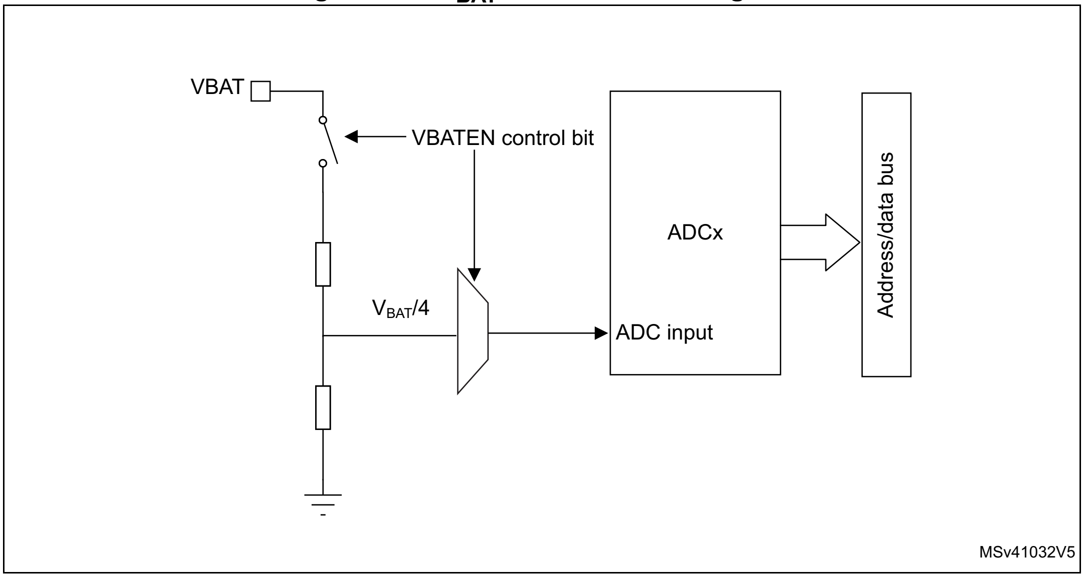
The VBATEN bit of the ADC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to an ADC input channel (refer to *Table ADC interconnection* in *Section 28.4.2: ADC pins and internal signals*).

As a consequence, the converted digital value is one fourth of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 245 shows the block diagram of the V_{BAT} sensing feature.

Figure 245. V_{BAT} channel block diagram



Note: The VBATEN bit in ADC_CCR must be configured to enable the conversion of the corresponding ADC internal channel.

28.4.35 Monitoring the internal voltage reference

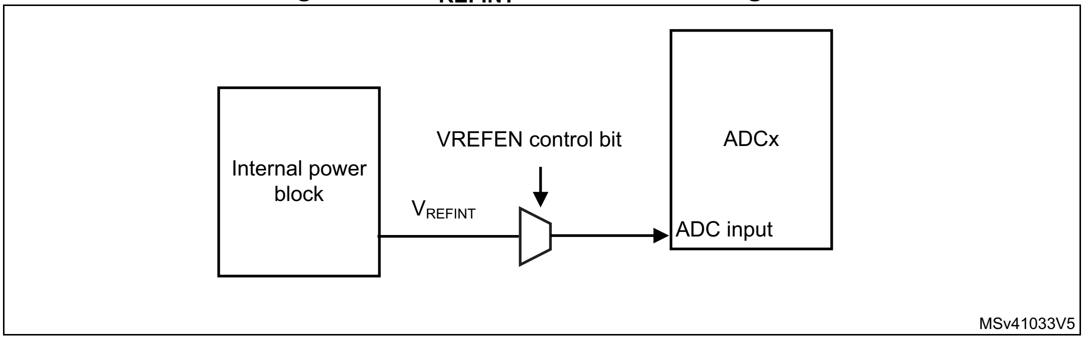
The internal voltage reference can be monitored to have a reference point for evaluating the ADC V_{REF+} voltage level.

The internal voltage reference is internally connected to an ADC input channel (refer to *Table ADC interconnection* in *Section 28.4.2: ADC pins and internal signals*).

The sampling time for this channel must be greater than the stabilization time specified in the product datasheet.

Figure 245 shows the block diagram of the V_{REFINT} sensing feature.

Figure 246. V_{REFINT} channel block diagram



Note: The VREFEN bit of the ADC_CCR register must be configured to enable the conversion of the corresponding ADC internal channel (V_{REFINT}).

Calculating the actual V_{DDA} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{DDA} = 3.3 V can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{DDA} voltage supplying the device:

$$V_{REF+} = 3.3\ V \times VREFINT\_CAL / VREFINT\_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For most applications V_{DDA} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{DDA}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{3.3\ V \times VREFINT\_CAL \times ADC\_DATA}{VREFINT\_DATA \times FULL\_SCALE}$$

For applications where V_{DDA} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{DDA}}{FULL\_SCALE} \times ADC\_DATA$$

Where:

- VREFINT_CAL is the VREFINT calibration value
- ADC_DATA is the value measured by the ADC on channel x (right-aligned)
- VREFINT_DATA is the actual VREFINT output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 16-bit resolution, it is 2^{16} - 1 = 65535 or with 8-bit resolution, 2^8 - 1 = 255.

Note: If ADC measurements are done using an output format other than 16-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

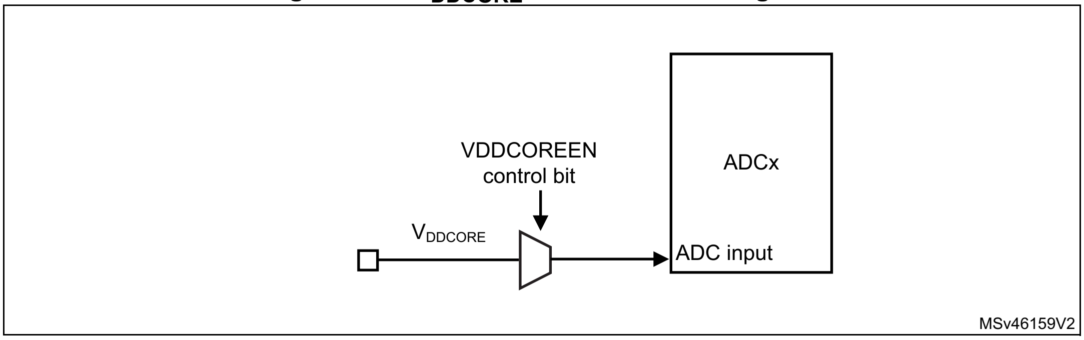
28.4.36 V_{DDCORE} supply voltage monitoring

V_{DDCORE} internal voltage can be monitored. V_{DDCORE} is internally connected to ADC2 V_{INP}[14] input channel.

The sampling time for this channel must be greater than the stabilization time specified in the product datasheet.

Figure 247 shows the block diagram of the V_{DDCORE} sensing feature.

Figure 247. V_{DDCORE} channel block diagram



Note: VDDCOREEN bit of ADC2_OR register must be set to enable ADC2 V_{INP}[14] conversion.

28.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (OVR flag)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 199. ADC interrupts per each ADC

Interrupt event	Event flag	Enable control bit
ADC ready	ADRDY	ADRDYIE
End of conversion of a regular group	EOC	EOCIE
End of sequence of conversions of a regular group	EOS	EOSIE
End of conversion of a injected group	JEOC	JEOCIE
End of sequence of conversions of an injected group	JEOS	JEOSIE
Analog watchdog 1 status bit is set	AWD1	AWD1IE
Analog watchdog 2 status bit is set	AWD2	AWD2IE
Analog watchdog 3 status bit is set	AWD3	AWD3IE
End of sampling phase	EOSMP	EOSMPIE

Table 199. ADC interrupts (continued)per each ADC

Interrupt event	Event flag	Enable control bit
Overrun	OVR	OVRIE
Injected context queue overflows	JQOVF	JQOVFIE

28.6 ADC registers (for each ADC)

Refer to *Section 1.2 on page 115* for a list of abbreviations used in register descriptions.

28.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
			r		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LDORDY**: ADC LDO output voltage ready bit

This bit is set and cleared by hardware. It indicates that the ADC internal LDO output is ready and that the ADC can be enabled or calibrated.

0: ADC LDO voltage regulator disabled

1: ADC LDO voltage regulator enabled

Note: Refer to Section 28.3: ADC implementation for the availability of the LDO regulator.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 28.4.22: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN=1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

28.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART is cleared to 0 (no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	LINCALRDYW6	LINCALRDYW5	LINCALRDYW4	LINCALRDYW3	LINCALRDYW2	LINCALRDYW1	Res.	Res.	Res.	Res.	Res.	ADCALLIN
rs	rw	rw	rw	rw	rw	rw	rw	rw	rw						rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOST	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
							rw			rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN=0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in single-ended inputs mode.

1: Writing ADCAL launches a calibration in differential inputs mode.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in deep-power-down mode.

0: ADC not in deep-power down

1: ADC in deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 28.4.6: ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bitfield only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 27 **LINCALRDYW6**: Linearity calibration ready Word 6

This control / status bit allows to read/write the 6th linearity calibration factor.

When the linearity calibration is complete, this bit is set. A bit clear launches the transfer of the linearity factor 6 into the LINCALFACT[29:0] of the ADC_CALFACT2 register. The bit is reset by hardware when the ADC_CALFACT2 register can be read (software must poll the bit until it is cleared).

When the *LINCALRDYW6 bit is reset, a new linearity factor 6 value can be written into* the LINCALFACT[29:0] of the ADC_CALFACT2 register. A bit set launches the linearity factor 6 update and the bit is effectively set by hardware once the update is done (software must poll the bit until it is set to indicate the write is effective).

Note: ADC_CALFACT2[29:10] *contains 0.* ADC_CALFACT2[9:0] *corresponds linearity correction factor bits[159:150].*

The software is allowed to toggle this bit only if the LINCALRDYW5, LINCALRDYW4, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged, see chapter 28.4.8: Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT) for details.

The software is allowed to update the linearity calibration factor by writing LINCALRDYWx only when ADEN=1 and ADSTART=0 and JADSTART=0 (ADC enabled and no conversion is ongoing)

Bit 26 **LINCALRDYW5**: Linearity calibration ready Word 5

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *corresponds linearity correction factor bits[149:120].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 25 **LINCALRDYW4**: Linearity calibration ready Word 4

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *correspond linearity correction factor bits[119:90].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW3, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 24 **LINCALRDYW3**: Linearity calibration ready Word 3

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *corresponds linearity correction factor bits[89:60].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW2 and LINCALRDYW1 bits are left unchanged.

Bit 23 **LINCALRDYW2**: Linearity calibration ready Word 2

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] *corresponds linearity correction factor bits[59:30].*

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW1 bits are left unchanged.

Bit 22 **LINCALRDYW1**: Linearity calibration ready Word 1

Refer to LINCALRDYW6 description.

Note: ADC_CALFACT2[29:0] corresponds linearity correction factor bits[29:0].

The software is allowed to toggle this bit only if the LINCALRDYW6, LINCALRDYW5, LINCALRDYW4, LINCALRDYW3 and LINCALRDYW2 bits are left unchanged.

Bits 21:17 Reserved, must be kept at reset value.

Bit 16 **ADCALLIN**: Linearity calibration

This bit is set and cleared by software to enable the Linearity calibration.

0: Writing ADCAL launches a calibration without the Linearity calibration.

1: Writing ADCAL launches a calibration with he Linearity calibration.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **BOOST**: Boost mode control

This bit is set and cleared by software to enable/disable the boost mode.

0: Boost mode off. Used when ADC clock < 20 MHz to save power at lower clock frequency.

1: Boost mode on. Must be used when ADC clock > 20 MHz.

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

When dual mode is enabled (bits DAMDF of ADC_CCR register are not equal to zero), the bit BOOST of the slave ADC is no more writable and its content must be equal to the master ADC BOOST bit.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set JADSTP only when JADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set ADSTP only when ADSTART=1 and ADDIS=0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

In dual ADC regular simultaneous mode and interleaved mode, the bit ADSTP of the master ADC must be used to stop regular conversions. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL=0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode (CONT=0, DISCEN=0) when software trigger is selected (EXTEN=0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- In discontinuous conversion mode (CONT=0, DISCEN=1), when the software trigger is selected (EXTEN=0x0): at the end of conversion (EOC) flag.
- in all other cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN=1 and ADDIS=0 (ADC is enabled and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO=1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN=1 and both ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

28.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					RES[2:0]			DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **JQDIS**: Injected Queue disable

These bits are set and cleared by software to disable the Injected Queue mechanism:

0: Injected Queue enabled

1: Injected Queue disabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

A set or reset of JQDIS bit causes the injected queue to be flushed and the ADC_JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel-0 monitored by AWD1

00001: ADC analog input channel-1 monitored by AWD1

.....

10010: ADC analog input channel-19 monitored by AWD1

others: Reserved, must not be used

Note: The channel selected by AWD1CH must be also selected into the SQi or JSQi bits.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no regular nor injected conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADC_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: ADC_JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: ADC_JSQR Mode 0: The Queue is never empty and maintains the last written configuration into ADC_JSQR.

1: ADC_JSQR Mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 28.4.22: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADC_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART=0 (which ensures that no injected conversion is ongoing).

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DAMDF of ADC_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADC_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADC_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADC_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN=1 and CONT=1.

The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DAMDF bits in ADC_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun Mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: Event 0

00001: Event 1

00010: Event 2

00011: Event 3

00100: Event 4

00101: Event 5

00110: Event 6

00111: Event 7

...

11111: Event 31

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 4:2 **RES[2:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

000: 16 bits

001: 14 bits

010: 12 bits

011: 10 bits

100: 8 bits

Others: Reserved, must not be used.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data Management configuration

This bit is set and cleared by software to select how ADC interface output data are managed.

00: Regular conversion data stored in DR only

01: DMA One Shot Mode selected

10: DFSDM mode selected

11: DMA Circular Mode selected

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bit DAMDF of the ADC_CCR register.

28.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OSVR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	RSHIFT4	RSHIFT3	RSHIFT2	RSHIFT1	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: Shift left 1-bit
0010: Shift left 2-bits
0011: Shift left 3-bits
0100: Shift left 4-bits
0101: Shift left 5-bits
0110: Shift left 6-bits
0111: Shift left 7-bits
1000: Shift left 8-bits
1001: Shift left 9-bits
1010: Shift left 10-bits
1011: Shift left 11-bits
1100: Shift left 12-bits
1101: Shift left 13-bits
1110: Shift left 14-bits
1111: Shift left 15-bits

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OSVR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RSHIFT4**: Right-shift data after Offset 4 correction

Refer to RSHIFT1 description.

Bit 13 **RSHIFT3**: Right-shift data after Offset 3 correction
Refer to RSHIFT1 description

Bit 12 **RSHIFT2**: Right-shift data after Offset 2 correction
Refer to RSHIFT1 description

Bit 11 **RSHIFT1**: Right-shift data after Offset 1 correction
This bitfield is set and cleared by software to right-shift 1-bit data after offset1 correction. This bit can only be used for 8-bit and 16-bit data format (see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)* for details).
0: Right-shifting disabled
1: Data is right-shifted 1-bit.

Bit 10 **ROVSM**: Regular Oversampling mode
This bit is set and cleared by software to select the regular oversampling mode.
0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)
1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)
Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular Oversampling
This bit is set and cleared by software to enable triggered oversampling
0: All oversampled conversions for a channel are done consecutively following a trigger
1: Each oversampled conversion for a channel needs a new trigger
Note: The software is allowed to write this bit only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling right shift
This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.
0000: No right shift
0001: Shift right 1-bit
0010: Shift right 2-bits
0011: Shift right 3-bits
0100: Shift right 4-bits
0101: Shift right 5-bits
0110: Shift right 6-bits
0111: Shift right 7-bits
1000: Shift right 8-bits
1001: Shift right 9-bits
1010: Shift right 10-bits
1011: Shift right 11-bits
Others: Reserved, must not be used.
Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no conversion is ongoing).

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **JOVSE**: Injected Oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected Oversampling disabled

1: Injected Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular Oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular Oversampling disabled

1: Regular Oversampling enabled

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing)

28.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles

001: 2.5 ADC clock cycles

010: 8.5 ADC clock cycles

011: 16.5 ADC clock cycles

100: 32.5 ADC clock cycles

101: 64.5 ADC clock cycles

110: 387.5 ADC clock cycles

111: 810.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 8.5 ADC clock cycles
011: 16.5 ADC clock cycles
100: 32.5 ADC clock cycles
101: 64.5 ADC clock cycles
110: 387.5 ADC clock cycles
111: 810.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL19	PCSEL18	PCSEL17	PCSEL16
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PCSEL15	PCSEL14	PCSEL13	PCSEL12	PCSEL11	PCSEL10	PCSEL9	PCSEL8	PCSEL7	PCSEL6	PCSEL5	PCSEL4	PCSEL3	PCSEL2	PCSEL1	PCSEL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 PCSEL[19:0] :Channel x ($V_{INP[i]}$) pre selection (x = 0 to 19)

These bits are written by software to pre select the input channel at IO instance to be converted.

0: Input Channel x (Vinp x) is not pre selected for conversion, the ADC conversion result with this channel shows wrong result.

1: Input Channel x (Vinp x) is pre selected for conversion

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.9 ADC watchdog threshold register 1 (ADC_LTR1)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTR1[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LTR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR1[25:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 28.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.10 ADC watchdog threshold register 1 (ADC_HTR1)

Address offset: 0x24

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HTR1[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HTR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR1[25:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 28.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 4th in the regular conversion sequence.

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion

0001: 2 conversions

...

1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

28.6.12 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 8th in the regular conversion sequence

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

28.6.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 11th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 10th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

28.6.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0..19) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

28.6.15 ADC regular Data Register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA[31:0]**: Regular Data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 28.4.27: Data management*.

28.6.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0..19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing) unless the context queue is enabled (JQDIS=0 in ADC_CFGR register).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS=0 (queue enabled), Hardware and software trigger detection disabled and If JQDIS=1 (queue disabled), Hardware trigger detection disabled (conversions can be launched by software

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

If JQM=1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 28.4.22: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: Event 0

00001: Event 1

00010: Event 2

00011: Event 3

00100: Event 4

00101: Event 5

00110: Event 6

00111: Event 7

...

11111: Event 31:

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART is cleared to 0 (no injected conversion is ongoing).

28.6.17 ADC injected channel y offset register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y-1), (y = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSATE	OFFSET_CH[4:0]					OFFSET[25:16]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SSATE:** Signed saturation Enable

This bit is written by software to enable or disable the Signed saturation feature.

This bit can be enabled only for 8-bit and 16-bit data format (see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)* for details).

0: Offset is subtracted maintaining data integrity and extending result size (9-bit and 17-bit signed format).

1: Offset is subtracted and result is saturated to maintain result size.

Note: The software is allowed to write this bit only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSET_CH[4:0]:** Channel selection for the Data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSETy[25:0] applies.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

Bits 25:0 **OFFSET[25:0]:** Data offset y for the channel programmed into bits OFFSETy_CH[4:0]

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the bits OFFSETy_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

When OFFSETy[25:0] bitfield is reset, the offset compensation is disabled.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

If several offset (OFFSETy) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

Ex: if OFFSET1_CH[4:0]=4 and OFFSET2_CH[4:0]=4, this is OFFSET1[25:0] which is subtracted when converting channel 4.

28.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y-1), (y= 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 28.4.27: Data management*.

28.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:16]			
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AWD2CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD2CH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel-i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel-i is monitored by AWD2

When AWD2CH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQi or JSQi bits.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD3CH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel-i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel-i is monitored by AWD3

When AWD3CH[19:0] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQi or JSQi bits.

The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.21 ADC watchdog lower threshold register 2 (ADC_LTR2)

Address offset: 0xB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTR2[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR2[25:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 28.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.22 ADC watchdog higher threshold register 2 (ADC_HTR2)

Address offset: 0xB4

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HTR2[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR2[25:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 28.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*.

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.23 ADC watchdog lower threshold register 3 (ADC_LTR3)

Address offset: 0xB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTR3[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **LTR3[25:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

Refer to *Section 28.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.24 ADC watchdog higher threshold register 3 (ADC_HTR3)

Address offset: 0xBC

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	HTR3[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **HTR3[25:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 28.4.30: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTRy, AWD_LTRy, AWDy)*

Note: The software is allowed to write these bits only when ADSTART=0 and JADSTART=0 (which ensures that no conversion is ongoing).

28.6.25 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0xC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIFSEL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]**: Differential mode for channels 19 to 0

These bits are set and cleared by software. They allow to select if a channel is configured as single ended or differential mode.

DIFSEL[i] = 0: ADC analog input channel-i is configured in single ended mode

DIFSEL[i] = 1: ADC analog input channel-i is configured in differential mode

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, JADSTP=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

28.6.26 ADC calibration factors register (ADC_CALFACT)

Address offset: 0xC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CALFACT_D[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CALFACT_S[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **CALFACT_D[10:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential conversion is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **CALFACT_S[10:0]**: Calibration Factors In single-Ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended conversion is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

28.6.27 ADC calibration factor register 2 (ADC_CALFACT2)

Address offset: 0xC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	LINCALFACT[29:16]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINCALFACT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **LINCALFACT[29:0]**: Linearity Calibration Factor

These bits are written by hardware or by software.

They hold 30-bit out of the 160-bit linearity calibration factor.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN=1, ADSTART=0 and JADSTART=0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

28.6.28 ADC2 option register (ADC2_OR)

Address offset: 0xD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDCOREEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **VDDCOREEN**: V_{DDCORE} enable bit

This bit can be set and cleared by software to control ADC2 channel 14.

0: V_{DDCORE} channel disabled

1: V_{DDCORE} channel enabled

28.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

28.7.1 ADC common status register (ADC_CSR)

Address offset: 0x00

Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

This register provides an image of the status bits of the different ADCs. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
					r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected Context Queue Overflow flag of the slave ADC
This bit is a copy of the JQOVF bit in the corresponding ADCx+1_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADCx+1_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADCx+1_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADCx+1_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADCx+1_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADCx+1_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADCx+1_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC
This bit is a copy of the EOS bit in the corresponding ADCx+1_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADCx+1_ISR register.

Bit 17 **EOSMP_SLV:** End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADCx+1_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADCx+1_ISR register.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

28.7.2 ADC common control register (ADC_CCR)

Address offset: 0x08

Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC[3:0]				CKMODE[1:0]	
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw			rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable

This bit is set and cleared by software to control V_{BAT} channel.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 23 **TSEN**: Temperature sensor voltage enable

This bit is set and cleared by software to control V_{SENSE} channel.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Note: The software is allowed to write this bit only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
Others: Reserved, must not be used

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: CK_ADCx (x=1 to 3) (Asynchronous clock mode), generated at product level (refer to *Section Reset and Clock Control (RCC)*)
01: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0xxx in RCC_CFGR register) and if the system clock has a 50% duty cycle.
10: adc_hclk/2 (Synchronous clock mode)
11: adc_hclk/4 (Synchronous clock mode)
In synchronous clock mode, there is no jitter in the delay from a timer trigger to the start of conversion.

Note: 2*The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).*

Bits 15:14 **DAMDF[1:0]:** Dual ADC mode data format

This bit-field is set and cleared by software. It specifies the data format in the common data register ADC_CDR.

00: Dual ADC mode without data packing (ADC_CDR and ADC_CDR2 registers not used).
01: Reserved.
10: Data formatting mode for 32 down to 10-bit resolution
11: Data formatting mode for 8-bit resolution

Note: The software is allowed to write these bits only when ADSTART=0 (which ensures that no regular conversion is ongoing).

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 200* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode.
All the ADCs are independent:
00000: Independent mode
The configurations 00001 to 01001 correspond to the following operating modes: dual mode, master and slave ADCs working together:
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined Interleaved mode + injected simultaneous mode
00100: Reserved.
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
All other combinations are reserved and must not be programmed

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL=0, JADSTART=0, ADSTART=0, ADSTP=0, ADDIS=0 and ADEN=0).

Table 200. DELAY bits versus ADC resolution

DELAY bits	16-bit resolution	14-bit resolution	12-bit resolution	10-bit resolution	8-bit resolution
0000	1.5 * $T_{adc\_ker\_ck}$	1.5 * $T_{adc\_ker\_ck}$	1.5 * $T_{adc\_ker\_ck}$	1.5 * $T_{adc\_ker\_ck}$	1.5 * $T_{adc\_ker\_ck}$
0001	2.5 * $T_{adc\_ker\_ck}$	2.5 * $T_{adc\_ker\_ck}$	2.5 * $T_{adc\_ker\_ck}$	2.5 * $T_{adc\_ker\_ck}$	2.5 * $T_{adc\_ker\_ck}$
0010	3.5 * $T_{adc\_ker\_ck}$	3.5 * $T_{adc\_ker\_ck}$	3.5 * $T_{adc\_ker\_ck}$	3.5 * $T_{adc\_ker\_ck}$	3.5 * $T_{adc\_ker\_ck}$
0011	4.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$
0100	5.5 * $T_{adc\_ker\_ck}$	5.5 * $T_{adc\_ker\_ck}$	5.5 * $T_{adc\_ker\_ck}$	5.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$
0101	6.5 * $T_{adc\_ker\_ck}$	6.5 * $T_{adc\_ker\_ck}$	6.5 * $T_{adc\_ker\_ck}$	5.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$
0110	7.5 * $T_{adc\_ker\_ck}$	7.5 * $T_{adc\_ker\_ck}$	6.5 * $T_{adc\_ker\_c}$	5.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$
0111	8.5 * $T_{adc\_ker\_ck}$	7.5 * $T_{adc\_ker\_ck}$	6.5 * $T_{adc\_ker\_ck}$	5.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$
1000	8.5 * $T_{adc\_ker\_ck}$	7.5 * $T_{adc\_ker\_ck}$	6.5 * $T_{adc\_ker\_ck}$	5.5 * $T_{adc\_ker\_ck}$	4.5 * $T_{adc\_ker\_ck}$
others: reserved	-	-	-	-	-

28.7.3 ADC common regular data register for dual mode (ADC_CDR)

Address offset: 0x0C

Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_SLV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDATA_MST[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 28.4.32: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 28.4.32: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

28.7.4 ADC common regular data register for 32-bit dual mode (ADC_CDR2)

Address offset: 0x10

Reset value: 0x0000 0000

The address offset is relative to the master ADC base address + 0x300.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA_ALT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDATA_ALT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs

In dual mode, these bits alternatively contains the regular 32-bit data of the master and the slave ADC. Refer to *Section 28.4.32: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, RSHIFT, SSATE)*.

28.7.5 ADC hardware configuration register (ADC_HWCFGR0)

Address offset: 0xF0

Reset value: 0x0000 0082

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_SIZE[3:0]				ADC_NUM[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **FIFO_SIZE[3:0]**: FIFO size
1000: FIFO depth = 8 stages

Bits 3:0 **ADC_NUM[3:0]**: Number of supported ADCs
0010: Two ADCs implemented

28.7.6 ADC version register (ADC_VERR)

Address offset: 0xF4

Reset value: 0x0000 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
Major revision = 0x3

Bits 3:0 **MINREV[3:0]**: Minor revision
Minor revision = 0x1

28.7.7 ADC identification register (ADC_IPIDR)

Address offset: 0xF8

Reset value: 0x0011 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Peripheral identifier
These bits returns the ADC identifier.
ID[31:0]=0x0011 0005

28.7.8 ADC size identification register (ADC_SIDR)

Address offset: 0xFC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size Identification
These bits returns the size of the memory region allocated to ADC registers.
The side of the memory region is of 1 Kbyte.
0xA3C5 DD01 (1 Kbyte)

28.8 ADC register map

The following table summarizes the ADC registers.

Table 201. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LDORDY	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																				0		0	0	0	0	0	0	0	0	0	0	0
0x04	**ADC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Table 201. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	**ADC_CR**	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	LINCALRDYW6	LINCALRDYW5	LINCALRDYW4	LINCALRDYW3	LINCALRDYW2	LINCALRDYW1	Res.	Res.	Res.	Res.	Res.	ADCALLIN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOST	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0	0	0	0	0	0	0						0								0			0	0	0	0	0	0
0x0C	**ADC_CFGR**	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM [2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL [4:0]					RES [2:0]			DMNGT [1:0]	
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADC_CFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	OSVR[9:0]										Res.	RSHIFT4	RSHIFT3	RSHIFT2	RSHIFT1	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	JOVSE	ROVSE
	Reset value							0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0				0	0
0x14	**ADC_SMPR1**	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_SMPR2**	Res.	Res.	SMP19 [2:0]			SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [2:0]		
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**ADC_PCSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL19	PCSEL18	PCSEL17	PCSEL16	PCSEL15	PCSEL14	PCSEL13	PCSEL12	PCSEL11	PCSEL10	PCSEL9	PCSEL8	PCSEL7	PCSEL6	PCSEL5	PCSEL4	PCSEL3	PCSEL2	PCSEL1	PCSEL0
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**ADC_LTR1**	Res.	Res.	Res.	Res.	Res.	Res.	LTR1[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_HTR1**	Res.	Res.	Res.	Res.	Res.	Res.	HTR1[25:0]																									
	Reset value							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x28	Reserved	Res.																															
0x2C	Reserved	Res.																															
0x30	**ADC_SQR1**	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x34	**ADC_SQR2**	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	SQ5[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x38	**ADC_SQR3**	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	SQ10[4:0]				
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0
0x3C	**ADC_SQR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	**ADC_DR**	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	Reserved	Res.																															
0x4C	**ADC_JSQR**	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50-0x5C	Reserved	Res.																															

Table 201. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	ADC_OFR1	SSATE	OFFSET1_CH[4:0]					OFFSET1[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	ADC_OFR2	SSATE	OFFSET2_CH[4:0]					OFFSET2[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	ADC_OFR3	SSATE	OFFSET3_CH[4:0]					OFFSET3[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	ADC_OFR4	SSATE	OFFSET4_CH[4:0]					OFFSET4[25:0]																									
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70-0x7C	Reserved	Res.																															
0x80	ADC_JDR1	JDATA1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	ADC_JDR2	JDATA2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	ADC_JDR3	JDATA3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	ADC_JDR4	JDATA4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90-0x9C	Reserved	Res.																															
0xA0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8-0xAC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xB0	ADC_LTR2	Res.	Res.	Res.	Res.	Res.	Res.	LTR2[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	ADC_HTR2	Res.	Res.	Res.	Res.	Res.	Res.	HTR2[25:0]																									
	Reset value							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 201. ADC register map and reset values for each ADC (offset=0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB8	**ADC_LTR3**	Res.	Res.	Res.	Res.	Res.	Res.	LTR3[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	**ADC_HTR3**	Res.	Res.	Res.	Res.	Res.	Res.	HTR3[25:0]																									
	Reset value							1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xC0	**ADC_DIFSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC4	**ADC_CALFACT**	Res.	Res.	Res.	Res.	Res.	CALFACT_D[10:0]											Res.	Res.	Res.	Res.	Res.	CALFACT_S[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xC8	**ADC_ CALFACT2**	Res.	Res.	LINCALFACT[29:0]																													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xD0	ADC2_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDCOREEN
																																	0

Table 202. ADC register map and reset values (master and slave ADC common registers) offset =0x300)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_CSR**	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							slave ADC2																master ADC1										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x04	Reserved	Res.																															
0x08	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC[3:0]				CKMODE[1:0]		DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0	0	0	0	0	0	0	0	0			0	0	0	0				0	0	0	0	0
0x0C	**ADC_CDR**	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADC_CDR2**	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF0	ADC_HWCFGR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFO_SIZE				ADC_NUM			
	Reset value																									1	0	0	0	0	0	1	0
0xF4	ADC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	1	0	0	0	1
0xF8	ADC_IPDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0xFC	ADC_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

29 Digital temperature sensor (DTS)

29.1 Introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

29.2 DTS main features

The temperature sensor block main features are the following:

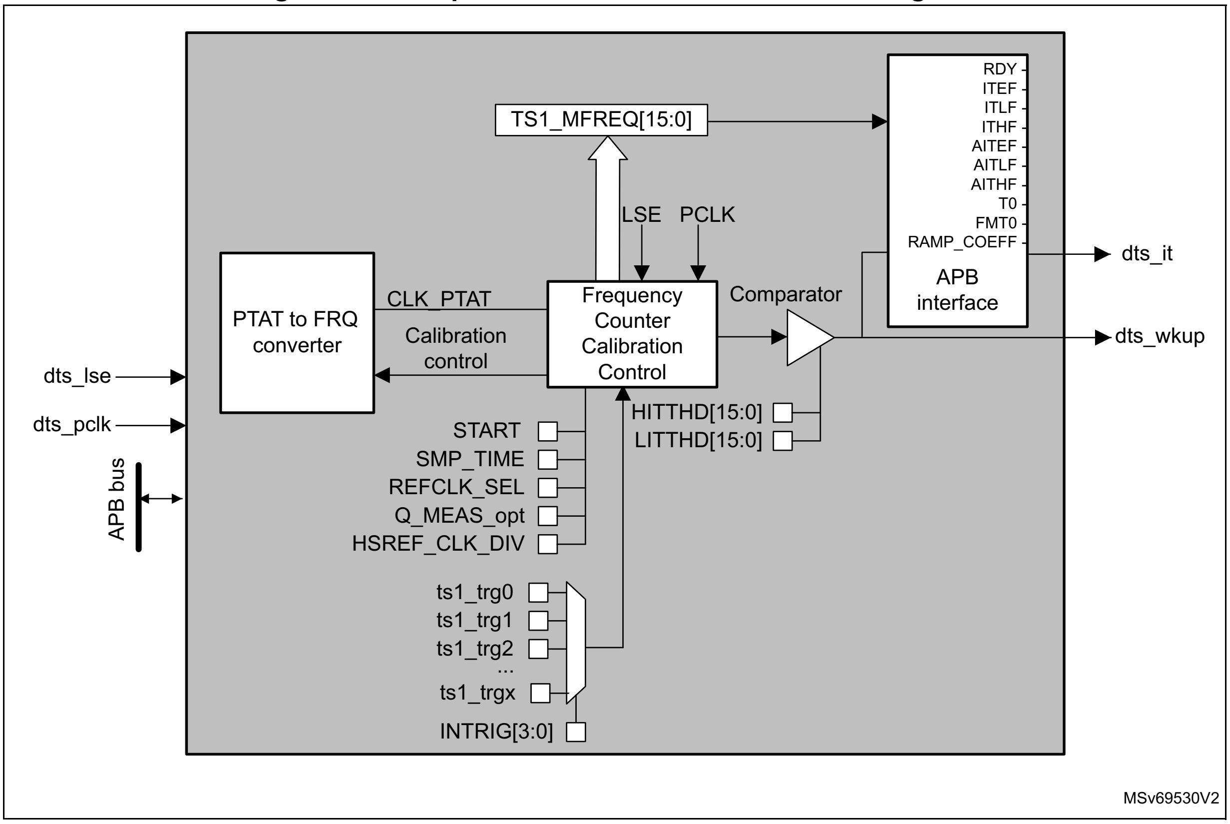
- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wakeup signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock

29.3 DTS functional description

29.3.1 DTS block diagram

The temperature sensor block diagram is shown in *Figure 248*.

Figure 248. Temperature sensor functional block diagram



29.3.2 DTS internal signals

Table 203. DTS internal input/output signals

Signal name	Signal type	Description
dts_lse	Digital input	LSE clock
dts_pclk	Digital input	APB clock
dts_it	Digital output	Temperature sensor interrupt
dts_wkup	Digital output	Temperature sensor wakeup

29.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

29.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in *Temperature sensor configuration register 1 (DTS_CFGR1)*:

- PCLK only (REFCLK_SEL = 0)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock
- PCLK and LSE (REFCLK_SEL = 1)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.
- LSE only (REFCLK_SEL = 1) and PCLK OFF

  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

29.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

29.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.

29.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

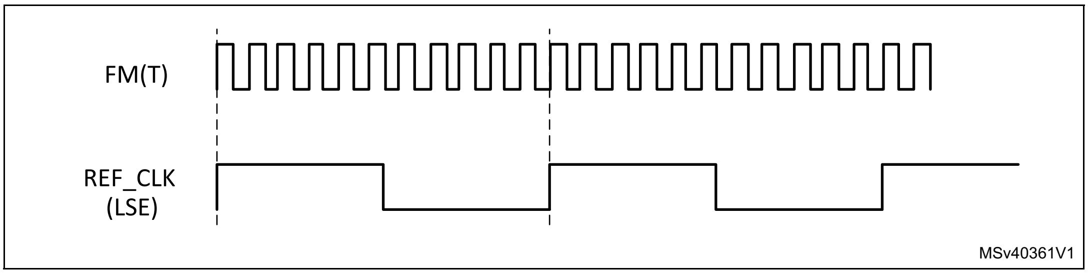
Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

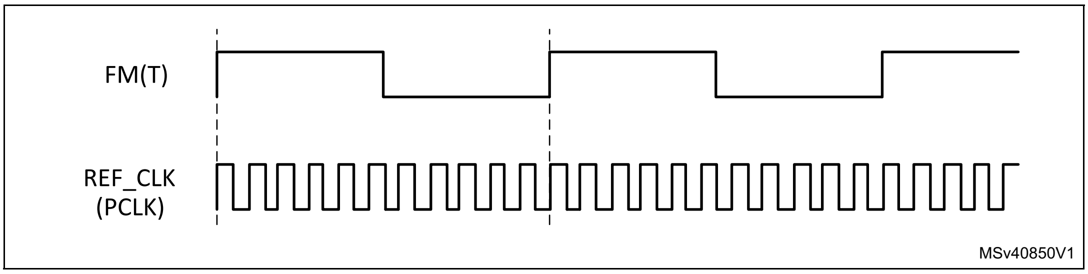
This counter behavior is shown in *Figure 249* and *Figure 250*.

Figure 249. Method for low REF_CLK frequencies



1. To increase the precision, FM(T) measurement can be done on several LSE periods.

Figure 250. Method for high REF_CLK frequencies



1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see *Temperature sensor data register (DTS_DR)*.

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

$$T = T_0 + ((F_{PCLK} / \text{TS1\_MFREQ}) \times \text{TS1\_SMP\_TIME} - 100 \times \text{TS1\_FMT0}) / \text{TS1\_RAMP\_COEFF}$$

where

T0 (factory calibration temperature) is equal to 30 °C.

TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.

TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

$$T = T_0 + ((F_{LSE} \times TS1\_MFREQ / TS1\_SMP\_TIME) - (100 \times TS1\_FMT0)) / TS1\_RAMP\_COEFF$$

29.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see *Table 204*).

Table 204. Sampling time configuration

TS1_SMP_TIME[3:0]	LSE or FM(T) clock cycle(s)
0000	1
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

29.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.

29.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFGR1.

Table 205. Trigger configuration

Name	TS1_INTRIG[3:0]				Comment
N.A	0	0	0	0	No hardware trigger
ts1_trg0	0	0	0	1	lptim1_out
ts1_trg1	0	0	1	0	lptim2_out
ts1_trg2	0	0	1	1	lptim3_out
ts1_trg3	0	1	0	0	exti13
ts1_trg4	0	1	0	1	Reserved
ts1_trg5	0	1	1	0	
ts1_trg6	0	1	1	1	
ts1_trg7	1	0	0	0	
ts1_trg8	1	0	0	1	
ts1_trg9	1	0	1	0	
ts1_trg10	1	0	1	1	
ts1_trg11	1	1	0	0	
ts1_trg12	1	1	0	1	
ts1_trg13	1	1	1	0	
ts1_trg14	1	1	1	1	

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 29.3.11: On-off control and ready flag), otherwise the trigger is ignored.

If a trigger source changes on-the-fly, the new trigger source signal should be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

29.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFGR1 register. The TS1_RDY flag in the *Temperature sensor status register (DTS_SR)* indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.

29.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to ‘0000’ in DTS_CFGR1.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFGR1 starts the measurement.

If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.

If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag become 1 and the measurement restarts.

Hardware trigger

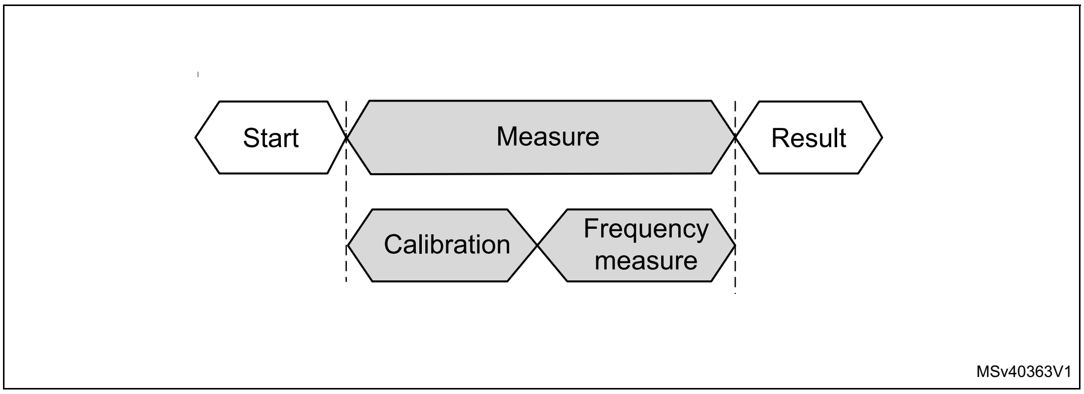
TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see *Section 29.3.5: Calibration*). The measurement period depends on the following DTS_CFGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 251. Temperature sensor sequence



29.4 DTS low-power modes

Table 206. Temperature sensor behavior in low-power modes

Mode	Description
Sleep	Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.
Stop	Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.

29.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source.The DTS interrupt line can be connected to the EXTI controller (see *Section 29.5.3: Asynchronous wakeup*) or to the CPU NVIC (see *Section 29.5.2: Synchronous interrupt*).

29.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ ITLF, TS1_ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 29.6.6*).

29.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 29.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 29.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 29.6.8*).

29.5.3 Asynchronous wakeup

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 29.5.1: Temperature window comparator*).

Three asynchronous wakeup events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.

The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits an the their description.

Table 207. Interrupt control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit from Sleep mode	Synchronous/ Asynchronous
At the end of measurement	TS1_ITEF in DTS_SR	TS1_ITEEN in DTS_ITENR	TS1_CITEF in DTS_ICIFR	NO	Synchronous on PCLK
When the measure is equal or exceeds the low threshold	TS1_ITLF in DTS_SR	TS1_ITLEN in DTS_ITENR	TS1_CITLF in DTS_ICIFR	NO	
When the measure is equal or exceeds the high threshold	TS1_ITHF in DTS_SR	TS1_ITHEN in DTS_ITENR	TS1_CITHF in DTS_ICIFR	NO	
At the end of measurement	TS1_AITEF in DTS_SR	TS1_AITEEN in DTS_ITENR	TS1_CAITEF in DTS_ICIFR	YES	Asynchronous
When the measure is equal or exceeds the low threshold	TS1_AITLF in DTS_SR	TS1_AITLEN in DTS_ITENR	TS1_CAITLF in DTS_ICIFR	YES	
When the measure is equal or exceeds the high threshold	TS1_AITHF in DTS_SR	TS1_AITHEN in DTS_ITENR	TS1_CAITHF in DTS_ICIFR	YES	

29.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

29.6.1 Temperature sensor configuration register 1 (DTS_CFGR1)

DTS_CFGR1 is the configuration register for temperature sensor 1.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSREF_CLK_DIV[6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_SMP_TIME[3:0]			
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TS1_INTRIG_SEL[3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
				rw	rw	rw	rw				rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSREF_CLK_DIV[6:0]**: High speed clock division ratio

These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).

0000000: No divider

0000001: No divider

0000010: 1/2 division ratio

...

1111111: 1/127 division ratio

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **Q_MEAS_OPT**: Quick measurement option bit

This bit is set and cleared by software. It is used to increase the measurement speed by suppressing the calibration step. It is effective only when the LSE clock is used as reference clock (REFCLK_SEL=1).

0: Measurement with calibration

1: Measurement without calibration

Bit 20 **REFCLK_SEL**: Reference clock selection bit

This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).

0: High speed reference clock (PCLK)

1: Low speed reference clock (LSE)

Bits 19:16 **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1

These bits allow increasing the sampling time to improve measurement precision.

When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement is performed at TS1_SMP_TIME period of CLK_PTAT.

When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is performed at TS1_SMP_TIME period of LSE.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1

These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Section 29.3.10: Trigger input*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TS1_START:** Start frequency measurement on temperature sensor 1

This bit is set and cleared by software.

0: No software trigger.

1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TS1_EN:** Temperature sensor 1 enable bit

This bit is set and cleared by software.

0: Temperature sensor 1 disabled

1: Temperature sensor 1 enabled

Note: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag is set when the sensor is ready.

29.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08

Reset value: 0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]	
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS1_FMT0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
00: 30 °C
01: 130 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1
This value is expressed in 0.1 kHz.

29.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS1_RAMP_COEFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1.
This value is expressed in Hz/°C.

29.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS1_HITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_LITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TS1_HITTHD[15:0]:** High interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]:** Low interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

29.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_MFREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

29.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
r									r	r	r		r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TS1_RDY:** Temperature sensor 1 ready flag

This bit is set and reset by hardware.
It indicates that a measurement is ongoing.
0: Temperature sensor 1 busy
1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHF:** Asynchronous interrupt flag for high threshold on temperature sensor 1

This bit is set by hardware when the high threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITHFEN bit is set

Bit 5 **TS1_AITLF:** Asynchronous interrupt flag for low threshold on temperature sensor 1

This bit is set by hardware when the low threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITLFEN bit is set

Bit 4 **TS1_AITEF:** Asynchronous interrupt flag for end of measure on temperature sensor 1

This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
0: End of measure not detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_AITEFEN bit is set

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHF:** Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK
This bit is set by hardware when the high threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITHFEN bit is set

Bit 1 **TS1_ITLF:** Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when the low threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITLFEN bit is set

Bit 0 **TS1_ITEF:** Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
0: No end of measurement detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_ITEFEN bit is set

29.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ AITHEN	TS1_ AITLEN	TS1_ AITEEN	Res.	TS1_ ITHEN	TS1_ ITLEN	TS1_ ITEEN
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHEN:** Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is above the high threshold (only when REFCLK_SEL= 1”)
0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1_AITLEN:** Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is below the low threshold (only when REFCLK_SEL= 1)
0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
1: Asynchronous interrupt on low threshold enabled for temperature sensor 1

Bit 4 **TS1_AITEEN:** Asynchronous interrupt enable flag for end of measurement on temperature sensor 1

This bit are set and cleared by software.

It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).

0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1

1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN:** Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.

It enables the interrupt when the measure reaches or is above the high threshold.

0: Synchronous interrupt for high threshold disabled on temperature sensor 1

1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN:** Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.

It enables the synchronous interrupt when the measure reaches or is below the low threshold.

0: Synchronous interrupt for low threshold disabled on temperature sensor 1

1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN:** Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.

It enables the synchronous interrupt for end of measurement.

0: Synchronous interrupt for end of measurement disabled on temperature sensor 1

1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

29.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ CAITHF	TS1_ CAITLF	TS1_ CAITEF	Res.	TS1_ CITHF	TS1_ CITLF	TS1_ CITEF
									rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF:** Asynchronous interrupt clear flag for high threshold on temperature sensor 1

Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF:** Asynchronous interrupt clear flag for low threshold on temperature sensor 1

Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.

Bit 4 **TS1_CAITEF:** Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.
Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_CITHF:** Interrupt clear flag for high threshold on temperature sensor 1
Writing this bit to 1 clears the TS1_ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF:** Interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF:** Interrupt clear flag for end of measurement on temperature sensor 1
Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

29.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS_OP31	TS_OP30	TS_OP29	TS_OP28	TS_OP27	TS_OP26	TS_OP25	TS_OP24	TS_OP23	TS_OP22	TS_OP21	TS_OP20	TS_OP19	TS_OP18	TS_OP17	TS_OP16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TS_OP15	TS_OP14	TS_OP13	TS_OP12	TS_OP11	TS_OP10	TS_OP9	TS_OP8	TS_OP7	TS_OP6	TS_OP5	TS_OP4	TS_OP3	TS_OP2	TS_OP1	TS_OP0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TS_OP[31:0]:** general purpose option bits

29.6.10 DTS register map

The following table summarizes the temperature sensor registers.

Table 208. DTS register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DTS_CFGR1	Res.	HSREF_CLK_DIV [6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_SMP_TIME [3:0]				Res.	Res.	Res.	Res.	TS1_INTRIG_SEL [3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0					0	0	0	0				0				0
0x04	Reserved																																
0x08	DTS_T0VALR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]		TS1_FMT0[15:0]															
	Reset value															X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	Reserved																																
0x10	DTS_RAMPVALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RAMP_COEFF[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	DTS_ITR1	TS1_HITTHD[15:0]																TS1_LITTHD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	Reserved																																
0x1C	DTS_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_MFREQ[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DTS_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
	Reset value																	0									0	0	0		0	0	0
0x24	DTS_ITENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
	Reset value																										0	0	0		0	0	0
0x28	DTS_ICIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_CAITHF	TS1_CAITLF	TS1_CAITEF	Res.	TS1_CITHF	TS1_CITLF	TS1_CITEF
	Reset value																										0	0	0		0	0	0
0x2C	DTS_OR	TS_OP31	TS_OP30	TS_OP29	TS_OP28	TS_OP27	TS_OP26	TS_OP25	TS_OP24	TS_OP23	TS_OP22	TS_OP21	TS_OP20	TS_OP19	TS_OP18	TS_OP17	TS_OP16	TS_OP15	TS_OP14	TS_OP13	TS_OP12	TS_OP11	TS_OP10	TS_OP9	TS_OP8	TS_OP7	TS_OP6	TS_OP5	TS_OP4	TS_OP3	TS_OP2	TS_OP1	TS_OP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

30 Digital-to-analog converter (DAC)

30.1 Introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DACx_OUTy pin can be used as general purpose input/output (GPIO) when the DAC output is disconnected from output pad and connected to on chip peripheral. The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

30.2 DAC main features

The DAC main features are the following (see *Figure 252: Dual-channel DAC block diagram*)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Each DAC output can be disconnected from the DACx_OUTy output pin
- DAC output connection to on-chip peripherals
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREF+ pin or internal VREFBUF reference

Figure 252 shows the block diagram of a DAC channel and *Table 210* gives the pin description.

30.3 DAC implementation

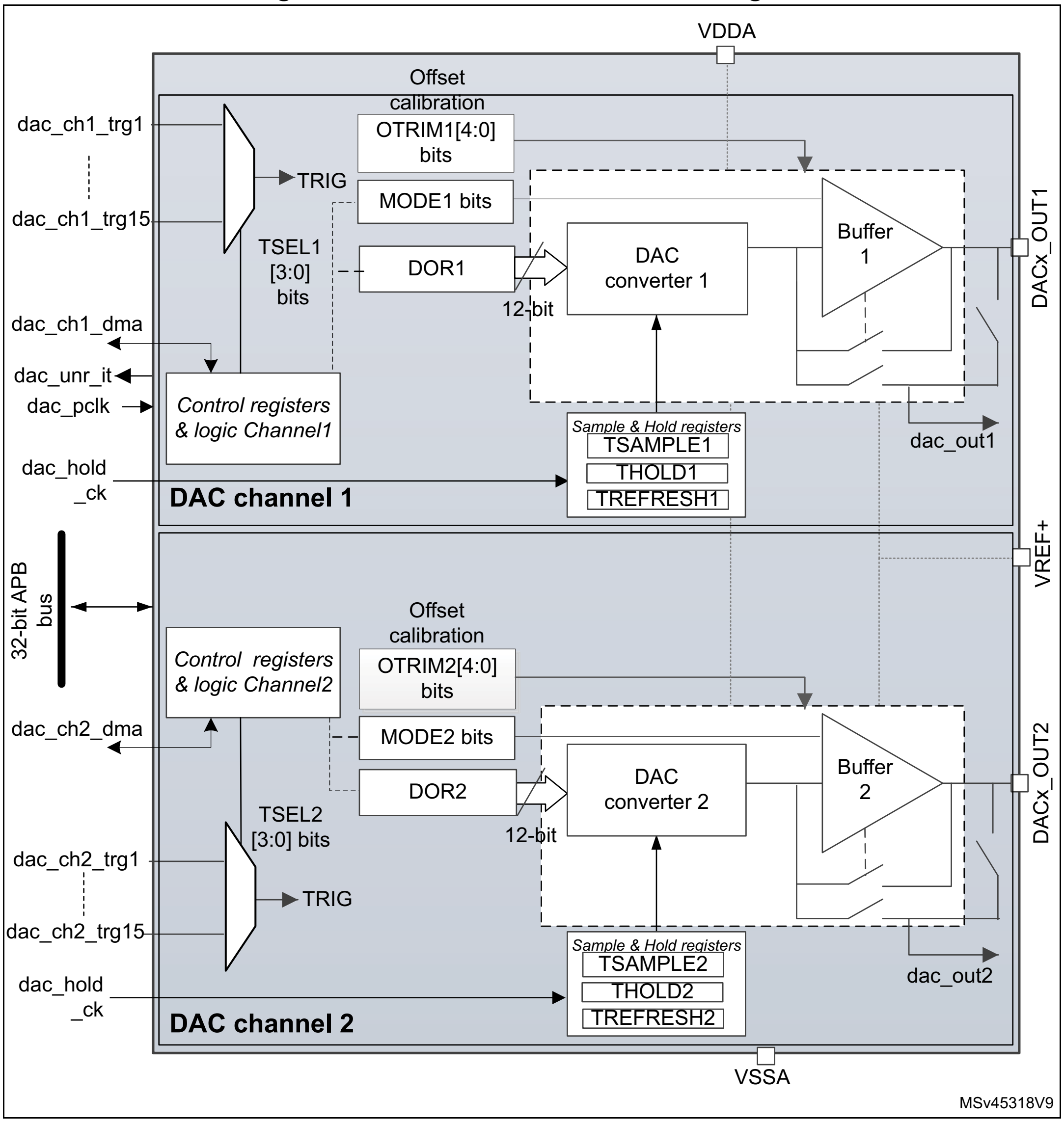
Table 209. DAC features

DAC features	DAC1
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1 Msps
Autonomous mode	-
VREF+ pin	X

30.4 DAC functional description

30.4.1 DAC block diagram

Figure 252. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. Refer to *Section 30.3: DAC implementation* for channel2 availability.

30.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- The DACx_OUTy can be disconnected from the output pin and used as an ordinary GPIO
- The dac_outx can use an internal pin connection to on-chip peripherals such as comparator, operational amplifier and ADC (if available).
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI clock source (dac_hold_ck).

Table 210. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 211. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_pclk	Input	DAC peripheral clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode
dac_out1	Analog output	DAC channel1 output for on-chip peripherals
dac_out2	Analog output	DAC channel2 output for on-chip peripherals

Table 212. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi (selected in the RCC)	LSI clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	tim1_trgo	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	tim2_trgo	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	tim4_trgo	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	tim5_trgo	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	tim6_trgo	Internal signal from on-chip timers
dac_chx_trg6 (x = 1, 2)	tim7_trgo	Internal signal from on-chip timers
dac_chx_trg7 (x = 1, 2)	tim8_trgo	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	tim15_trgo	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	lptim1_out	Internal signal from on-chip timers
dac_chx_trg12(x = 1, 2)	lptim2_out	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	exti9	External pin

30.4.3 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

30.4.4 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

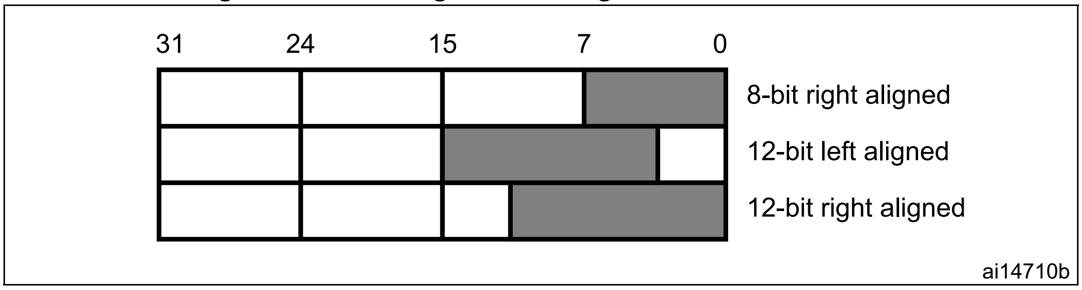
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHRx register is then loaded into the DAC_DORx register either automatically, by software trigger or by an external event trigger.

Figure 253. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

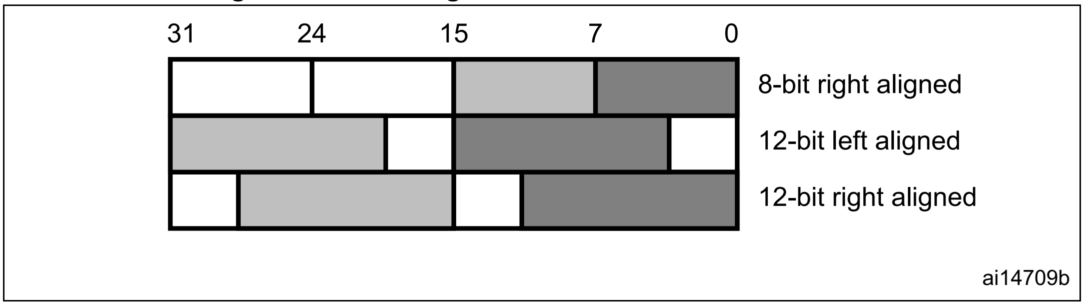
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 254. Data registers in dual DAC channel mode



30.4.5 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_pclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_pclk clock cycles after the trigger signal.

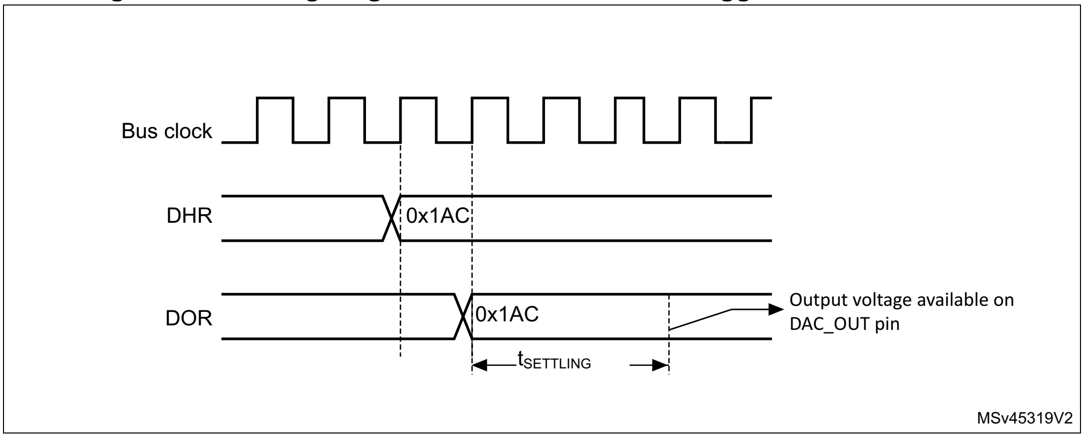
When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

HFSEL bit of DAC_CR must be set when dac_pclk clock speed is faster than 80 MHz. It adds an extra delay of three dac_pclk clock cycles to the transfer from DAC_DHRx register to DAC_DORx register ($t_{SETTLING}$).

The DAC_DORx update rate is limited to 1/3 of dac_pclk clock frequency. When HFSEL bit is set, this rate is limited to 1/8 of the dac_pclk clock frequency.

When HFSEL is set, it is not allowed to write to the DAC_DHRx register during a period of eight clock cycles after the ENx bit is set. During this period, making software/hardware triggering is not allowed either.

Figure 255. Timing diagram for conversion with trigger disabled TEN = 0



30.4.6 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DAC output} = V_{REF} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

30.4.7 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in *Section 30.4.2*.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_pclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_pclk clock cycle.

30.4.8 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

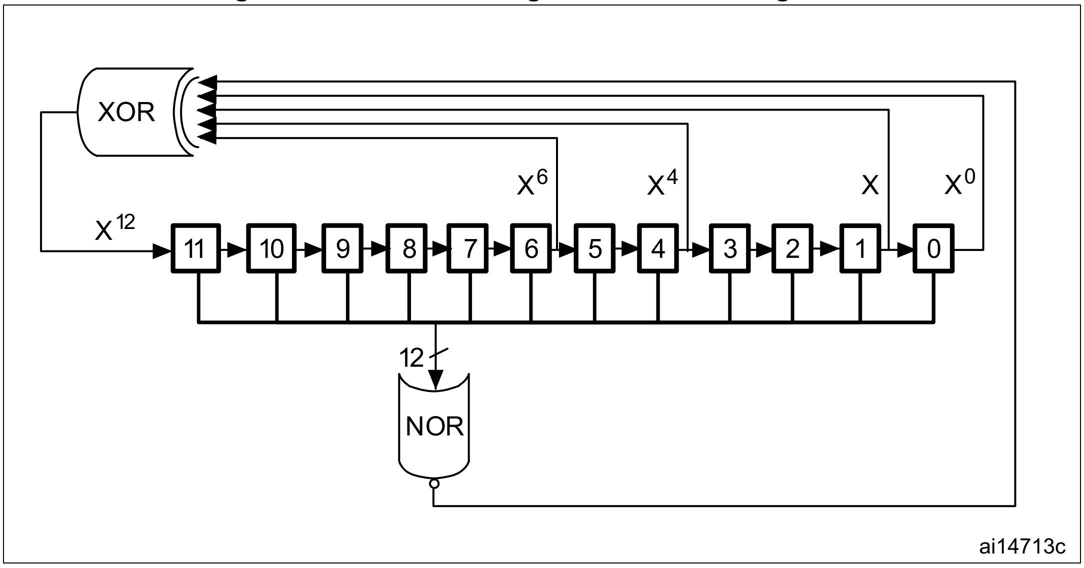
The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

30.4.9 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_pclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 256. DAC LFSR register calculation algorithm

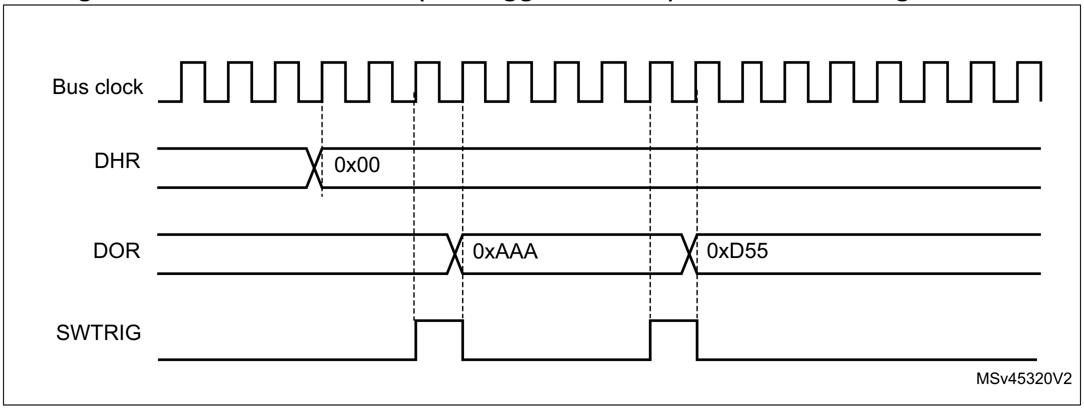


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 257. DAC conversion (SW trigger enabled) with LFSR wave generation



Note: The DAC trigger must be enabled for noise generation by setting the TENx bit in the DAC_CR register.

30.4.10 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_pclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 258. DAC triangle wave generation

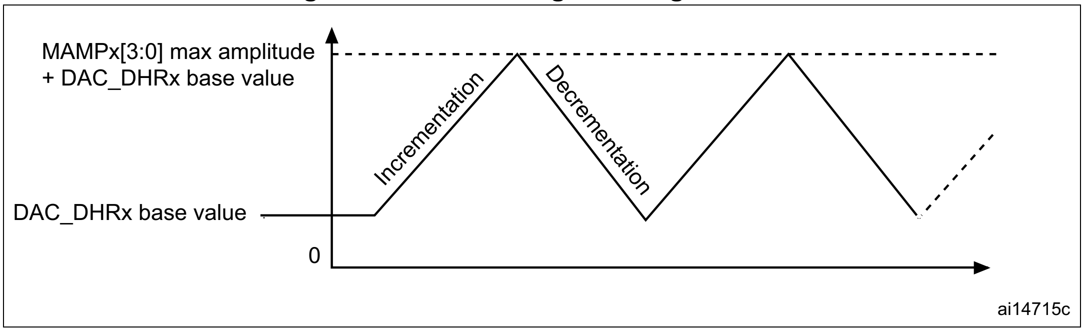
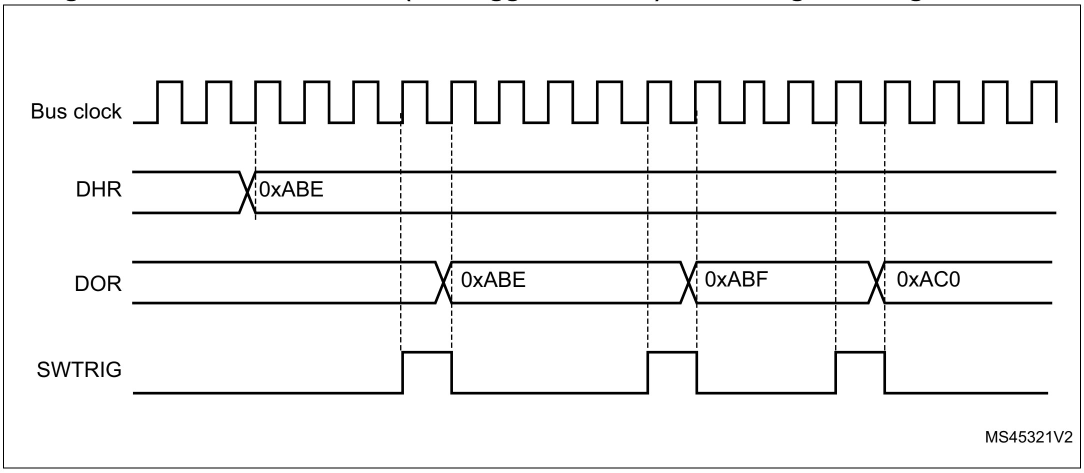


Figure 259. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

30.4.11 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin
- 001: DAC is connected to external pin and to on-chip peripherals

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin
- 011: DAC is connected to on-chip peripherals

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI low-speed clock (dac_hold_ck) in addition to the dac_pclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (APB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11.
- 62 cycles are required for hold phase: THOLDx[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 213. Sample and refresh timings

Buffer State	t_{SAMP}(1)(2)	$t_{REFRESH}$(2)(3)
Enable	$7\ \mu s + (10 * R_{BON} * C_{SH})$	$7\ \mu s + (R_{BON} * C_{SH}) * \ln(2 * N_{LSB})$
Disable	$3\ \mu s + (10 * R_{BOFF} * C_{SH})$	$3\ \mu s + (R_{BOFF} * C_{SH}) * \ln(2 * N_{LSB})$

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{DDA} = 3.0 V

Sampling phase:

$t_{SAMP} = 7\ \mu s + (10 * 2000 * 100 * 10^{-9}) = 2.007$ ms
(where $R_{BON} = 2\ k\Omega$)

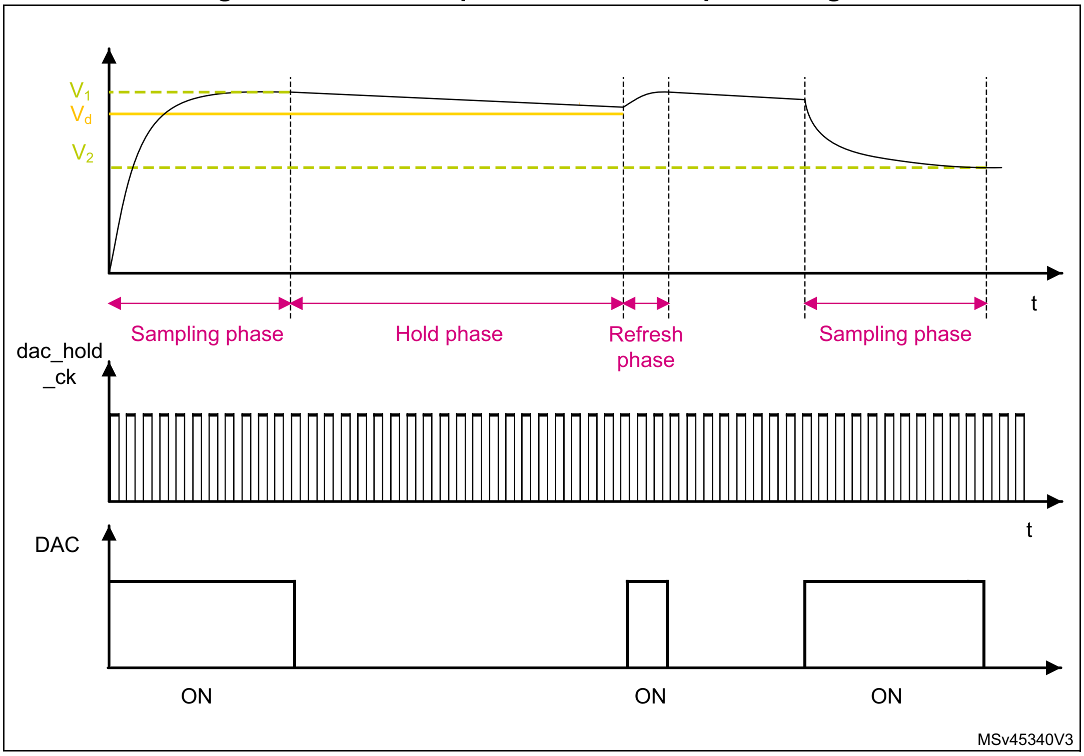
Refresh phase:

$t_{REFRESH} = 7\ \mu s + (2000 * 100 * 10^{-9}) * \ln(2*10) = 606.1\ \mu s$
(where N_{LSB} = 10 (10 LSB drop during the hold phase)

Hold phase:

$D_v = i_{leak} * t_{hold} / C_{SH} = 0.0073$ V (10 LSB of 12bit at 3 V)
i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)
$t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867$ ms

Figure 260. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin
- 101: DAC is connected to external pin and to on chip peripherals

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin and to on chip peripherals
- 111: DAC is connected to on chip peripherals
  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Table 214. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	0	1			Connected to external pin and to on chip-peripherals (such as comparators)
0	1	0		Disabled	Connected to external pin
0	1	1			Connected to on chip peripherals (such as comparators)

Table 214. Channel output modes summary (continued)

MODEx[2:0]			Mode	Buffer	Output connections
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	0	1			Connected to external pin and to on chip peripherals (such as comparators)
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)
1	1	1			Connected to on chip peripherals (such as comparators)

30.4.12 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = \left((D/\ 2^N) \times G \times V_{REF}\right) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REF} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and $V_{OS} = 0$.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to $V_{REF+}/2$ signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{REF+} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) Until the CAL_FLAGx is read as 1 or the maximum trimming code is reached, increment OTRIMx[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, V_{REF+} and temperature conditions do not change during device operation while it enters more often in Standby and VBAT modes, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

30.4.13 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHRx register is transferred into DAC_DORx (three dac_pclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSELx[3:0] and STINCTRIGSELx[3:0] bits.
2. Configure the DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHRx register and transfers it into DAC_DOR1 (three APB clock cycles later).

30.4.14 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

11 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DAC_DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_pclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_pclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_pclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_pclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_pclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three APB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_pclk clock cycles later). Then the DAC channel2 triangle counter is updated.

30.5 DAC in low-power modes

Table 215. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC can be used with DMA
Stop	The DAC remains active with a static output value if the sample and hold mode is selected using LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

30.6 DAC interrupts

Table 216. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

30.7 DAC registers

Refer to *Section 1 on page 115* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

30.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable
This bit is set and cleared by software.
0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled
Note: This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095
Note: These bits are available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable
These bits are set/reset by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection
These bits select the external event used to trigger DAC channel2
0000: SWTRIG2
0001: dac_ch2_trg1
0010: dac_ch2_trg2
...
1111: dac_ch2_trg15
Refer to the trigger selection tables in *Section 30.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger
0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_pclk clock cycle later to the DAC_DOR2 register
1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_pclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 reg*ister to the DAC_DOR2 register takes only one dac_pclk clock cycle.*

These bits are available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.
0: DAC channel2 disabled
1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 15 **HFSEL**: High frequency interface mode enable

This bit is set and cleared by software to enable/disable DAC interface high speed mode
This bit need to be set when dac_pclk clock frequency is higher than 80 MHz.
0: High frequency interface mode disabled
1: High frequency interface mode enabled

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.
0: DAC channel1 in normal operating mode
1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.
0: DAC channel1 DMA Underrun Interrupt disabled
1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.
0: DAC channel1 DMA mode disabled
1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable

These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection

These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in *Section 30.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.

Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable

This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_pclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_pclk clock cycles later to the DAC_DOR1 register

Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_pclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable

This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

30.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_pclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

30.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

30.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

30.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

30.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

30.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

30.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

30.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

30.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

30.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

30.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output

These bits are read-only, they contain data output for DAC channel1.

30.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output

These bits are read-only, they contain data output for DAC channel2.

30.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1													

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 30.3: DAC implementation.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bits 10:0 Reserved, must be kept at reset value.

30.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

30.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
													rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: DAC channel2 is connected to external pin and to on chip peripherals with buffer enabled

010: DAC channel2 is connected to external pin with buffer disabled

011: DAC channel2 is connected to on chip peripherals with Buffer disabled

– DAC channel2 in sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: DAC channel2 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel2 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel2 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN2 = 0.

Refer to Section 30.3: DAC implementation for the availability of DAC channel2.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel1 mode:

– DAC channel1 in normal mode

000: DAC channel1 is connected to external pin with Buffer enabled

001: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

010: DAC channel1 is connected to external pin with Buffer disabled

011: DAC channel1 is connected to on chip peripherals with Buffer disabled

– DAC channel1 in sample & hold mode

100: DAC channel1 is connected to external pin with Buffer enabled

101: DAC channel1 is connected to external pin and to on chip peripherals with Buffer enabled

110: DAC channel1 is connected to external pin and to on chip peripherals with Buffer disabled

111: DAC channel1 is connected to on chip peripherals with Buffer disabled

Note: This register can be modified only when EN1 = 0.

30.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x *LSI clock period.*

30.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

Note: It represents the number of LSI clocks to perform a sample phase. Sampling time = (TSAMPLE1[9:0] + 1) x *LSI clock period.*

30.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in sample and hold mode).
Hold time = (THOLD[9:0]) x LSI clock period
These bits are available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

30.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI clock period
These bits are available only on dual-channel DACs. Refer to *Section 30.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

30.7.21 DAC hardware configuration register (DAC_HWCFGR0)

Address offset: 0x3F0

Reset value: 0x0000 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.				Res.				OR_CFG[7:0]							
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SAMPLE[3:0]				TRIANGLE[3:0]				LFSR[3:0]				DUAL[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **OR_CFG[7:0]**: option register bit width
- 0000 0000: no option register
- 0000 0010: 2 bits
- 0001 0000: 16 bits
- 0010 0000: 32 bits
- Others: reserved

Bits 15:12 **SAMPLE[3:0]**: Sample and hold mode capability
- 0000: Sample and hold hardware not implemented
- 0001: Sample and hold hardware implemented
- Others: reserved

Bits 11:8 **TRIANGLE[3:0]**: Triangle wave generation capability
- 0000: triangle and wave generation hardware not implemented
- 0001: triangle and wave generation hardware implemented
- Others: reserved

Bits 7:4 **LFSR[3:0]**: Pseudonoise wave generation capability
- 0000: LFSR wave generation hardware not implemented
- 0001: LFSR wave generation hardware implemented
- Others: reserved

Bits 3:0 **DUAL[3:0]**: Dual DAC capability
- 0000: Single DAC digital interface implemented
- 0001: Dual DAC digital interface implemented
- Others: reserved

30.7.22 DAC version register (DAC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major IP revision
Major revision = 0x3

Bits 3:0 **MINREV[3:0]**: Minor IP revision
Major revision = 0x1

30.7.23 DAC identification register (DAC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0011 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification code
c7amba_dacif = 0x0011 0011

30.7.24 DAC size identification register (DAC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size and Identification register
0xA3C5 DD01

30.7.25 DAC register map

Table 217 summarizes the DAC registers.

Table 217. DAC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[3:1]			TSEL2[0]	TEN2	EN2	HFSEL	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:1]			TSEL1[0]	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	DAC_ SWTRGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	DAC_ DHR12R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	DAC_ DHR12L1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	DAC_ DHR8R1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	DAC_ DHR12R2	Res.	Res.	Res.	Res.	Res.												Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	DAC_ DHR12L2	Res.												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	DAC_ DHR8R2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DAC_ DHR12RD	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	DAC_ DHR12LD	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				

Table 217. DAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	DAC_DHR8RD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	DAC_DOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Res.	Res.	Res.	Res.	Res.												Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x34	DAC_SR	BWST2	CAL_FLAG2	DMAUDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0	0	0													
0x38	DAC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	DAC_MCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value														0	0	0														0	0	0
0x40	DAC_SHSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	DAC_SHSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	DAC_SHHR	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	DAC_SHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x50-0x54	Reserved	Res.																															
0x58-0x60	Reserved	Res.																															
0x64-0x68	Reserved	Res.																															
0x3F0	DAC_HWCFGR0	Res.								OR_CFG[7:0]								SAMPLE [3:0]				TRIANGLE [3:0]				LFSR[3:0]				DUAL[3:0]			
	Reset value	0x0000 1111																															

Table 217. DAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F4	**DAC_ VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value	0x0000 0031																															
0x3F8	**DAC_ IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x3FC	**DAC_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	0	1	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

31 Voltage reference buffer (VREFBUF)

31.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs and DACs, and also as voltage reference for external components through the VREF+ pin.

31.2 VREFBUF functional description

The internal voltage reference buffer supports four voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: around 2.5 V
- VRS = 001: around 2.048 V
- VRS = 010: around 1.8 V
- VRS = 011: around 1.5 V (ADC and DAC peripherals are not compatible with this reference voltage setting)

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 218. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

31.3 VREFBUF registers

31.3.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
									rw	rw	rw	r		rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

000: Voltage reference set to 2.5 V
001: Voltage reference set to 2.048 V
010: Voltage reference set to 1.8 V
011: Voltage reference set to 1.5 V
Others: Reserved

Note: The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.
1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.
0: V_{REF+} pin is internally connected to the voltage reference buffer output.
1: V_{REF+} pin is high impedance.
Refer to *Table 218: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.
0: Internal voltage reference mode disable (external voltage reference mode).
1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

31.3.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **TRIM[5:0]**: Trimming code

These bits are automatically initialized after reset with the trimming value stored in the OTP fuses during the production test.

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

31.3.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 219. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

32 Digital filter for sigma delta modulators (DFSDM)

32.1 Introduction

Digital filter for sigma delta modulators (DFSDM) is a high-performance module dedicated to interface external ΣΔ modulators. It is featuring up to 8 external digital serial interfaces (channels) and up to 6 digital filters with flexible Sigma Delta stream digital processing options to offer up to 24-bit final ADC resolution. DFSDM also features optional parallel data stream input from internal ADC peripherals or from device memory.

An external ΣΔ modulator provides digital data stream of converted analog values from the external ΣΔ modulator analog input. This digital data stream is sent into a DFSDM input channel through a serial interface. DFSDM supports several standards to connect various ΣΔ modulator outputs: SPI interface and Manchester coded 1-wire interface (both with adjustable parameters). DFSDM module supports the connection of up to 8 multiplexed input digital serial channels which are shared with up to 6 DFSDM modules. DFSDM module also supports alternative parallel data inputs from up to 8 internal 16-bit data channels (from internal ADCs or from device memory).

DFSDM is converting an input data stream into a final digital data word which represents an analog input value on a ΣΔ modulator analog input. The conversion is based on a configurable digital process: the digital filtering and decimation of the input serial data stream.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, length of filter, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion mode and continuous mode. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible timer triggering system can be used to control the start of conversion of DFSDM. This timing control is capable of triggering simultaneous conversions or inserting a programmable delay between conversions.

DFSDM features an analog watchdog function. Analog watchdog can be assigned to any of the input channel data stream or to final output data. Analog watchdog has its own digital filtering of input data stream to reach the required speed and resolution of watched data.

To detect short-circuit in control applications, there is a short-circuit detector. This block watches each input channel data stream for occurrence of stable data for a defined time duration (several 0’s or 1’s in an input data stream).

An extremes detector block watches final output data and stores maximum and minimum values from the output data values. The extremes values stored can be restarted by software.

Two power modes are supported: normal mode and stop mode.

32.2 DFSDM main features

- Up to 8 multiplexed input digital serial channels:
  - configurable SPI interface to connect various ΣΔ modulators
  - configurable Manchester coded 1 wire interface support
  - clock output for ΣΔ modulator(s)
- Alternative inputs from up to 8 internal digital parallel channels:
  - inputs with up to 16 bit resolution
  - internal sources: ADCs data or memory (CPU/DMA write) data streams
- Adjustable digital signal processing:
  - $Sinc^x$ filter: filter order/type (1..5), oversampling ratio (up to 1..1024)
  - integrator: oversampling ratio (1..256)
- Up to 24-bit output data resolution:
  - right bit-shifter on final data (0..31 bits)
- Signed output data format
- Automatic data offset correction (offset stored in register by user)
- Continuous or single conversion
- Start-of-conversion synchronization with:
  - software trigger
  - internal timers
  - external events
  - start-of-conversion synchronously with first DFSDM filter (DFSDM_FLT0)
- Analog watchdog feature:
  - low value and high value data threshold registers
  - own configurable $Sinc^x$ digital filter (order = 1..3, oversampling ratio = 1..32)
  - input from output data register or from one or more input digital serial channels
  - continuous monitoring independently from standard conversion
- Short-circuit detector to detect saturated analog input values (bottom and top ranges):
  - up to 8-bit counter to detect 1..256 consecutive 0’s or 1’s on input data stream
  - monitoring continuously each channel (8 serial channel transceiver outputs)
- Break generation on analog watchdog event or short-circuit detector event
- Extremes detector:
  - store minimum and maximum values of output data values
  - refreshed by software
- Pulse skipper feature to support beamforming applications (delay line like behavior)
- DMA may be used to read the conversion data
- Interrupts: end of conversion, overrun, analog watchdog, short-circuit, channel clock absence
- “regular” or “injected” conversions:
  - “regular” conversions can be requested at any time or even in continuous mode without having any impact on the timing of “injected” conversions
  - “injected” conversions for precise timing and with high conversion priority

32.3 DFSDM implementation

This section describes the configuration implemented in DFSDMx.

Table 220. DFSDM1 implementation

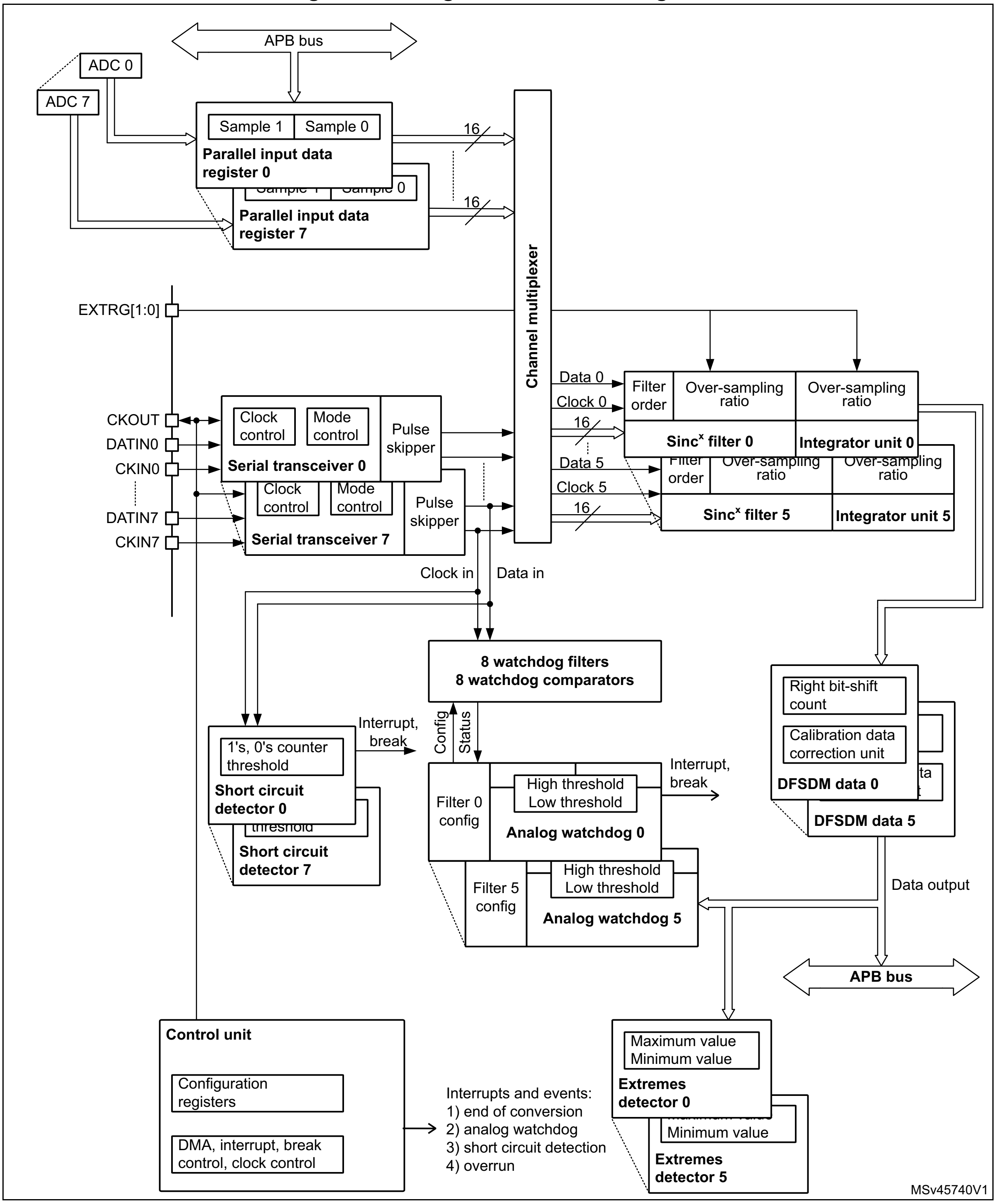
DFSDM features	DFSDM1
Number of channels	8
Number of filters	6
Input from internal ADC	X
Supported trigger sources	32(1)
Pulses skipper	X

1. Refer to *Table 223: DFSDM triggers connection* for available trigger sources.

32.4 DFSDM functional description

32.4.1 DFSDM block diagram

Figure 261. Single DFSDM block diagram



1. This example shows 6 DFSDM filters and 8 input channels (max. configuration).

32.4.2 DFSDM pins and internal signals

Table 221. DFSDM external pins

Name	Signal Type	Remarks
VDD	Power supply	Digital power supply.
VSS	Power supply	Digital ground power supply.
CKIN[7:0]	Clock input	Clock signal provided from external ΣΔ modulator. FT input.
DATIN[7:0]	Data input	Data signal provided from external ΣΔ modulator. FT input.
CKOUT	Clock output	Clock output to provide clock signal into external ΣΔ modulator.
EXTRG[1:0]	External trigger signal	Input trigger from two EXTI signals to start analog conversion (from GPIOs: EXTI11, EXTI15).

Table 222. DFSDM internal signals

Name	Signal Type	Remarks
dfsdm_jtrg[31:0]	Internal/ external trigger signal	Input trigger from internal/external trigger sources in order to start analog conversion (from internal sources: synchronous input, from external sources: asynchronous input with synchronization). See *Table 223* for details.
dfsdm_break[3:0]	break signal output	Break signals event generation from Analog watchdog or short-circuit detector
dfsdm_dma[5:0]	DMA request signal	DMA request signal from each DFSDM_FLTx (x=0..5): end of injected conversion event.
dfsdm_it[5:0]	Interrupt request signal	Interrupt signal for each DFSDM_FLTx (x=0..5)
dfsdm_dat_adc[15:0]	ADC input data	Up to 4 internal ADC data buses as parallel inputs.

Table 223. DFSDM triggers connection

Trigger name	Trigger source
dfsdm_jtrg0	TIM1_TRGO
dfsdm_jtrg1	TIM1_TRGO2
dfsdm_jtrg2	TIM8_TRGO
dfsdm_jtrg3	TIM8_TRGO2
dfsdm_jtrg4	TIM3_TRGO
dfsdm_jtrg5	TIM4_TRGO
dfsdm_jtrg6	TIM16_OC1
dfsdm_jtrg7	TIM6_TRGO
dfsdm_jtrg8	TIM7_TRGO

Table 223. DFSDM triggers connection (continued)

Trigger name	Trigger source
dfsdm_jtrg[23:9]	Reserved
dfsdm_jtrg24	EXTI11
dfsdm_jtrg25	EXTI15
dfsdm_jtrg26	LPTIMER1
dfsdm_jtrg27	LPTIMER2
dfsdm_jtrg28	LPTIMER3
dfsdm_jtrg[31:29]	Reserved

Table 224. DFSDM break connection

Break name	Break destination
dfsdm_break[0]	TIM1/TIM15 break
dfsdm_break[1]	TIM1 break2 / TIM16 break
dfsdm_break[2]	TIM8/TIM17 break
dfsdm_break[3]	TIM8 break2

32.4.3 DFSDM reset and clocks

DFSDM on-off control

The DFSDM interface is globally enabled by setting DFSDMEN=1 in the DFSDM_CH0CFGR1 register. Once DFSDM is globally enabled, all input channels (y=0..7) and digital filters DFSDM_FLTx (x=0..5) start to work if their enable bits are set (channel enable bit CHEN in DFSDM_CHyCFGR1 and DFSDM_FLTx enable bit DFEN in DFSDM_FLTxCR1).

Digital filter x DFSDM_FLTx (x=0..5) is enabled by setting DFEN=1 in the DFSDM_FLTxCR1 register. Once DFSDM_FLTx is enabled (DFEN=1), both Sinc^x digital filter unit and integrator unit are reinitialized.

By clearing DFEN, any conversion which may be in progress is immediately stopped and DFSDM_FLTx is put into stop mode. All register settings remain unchanged except DFSDM_FLTxAWSR and DFSDM_FLTxISR (which are reset).

Channel y (y=0..7) is enabled by setting CHEN=1 in the DFSDM_CHyCFGR1 register. Once the channel is enabled, it receives serial data from the external $\Sigma\Delta$ modulator or parallel internal data sources (ADCs or CPU/DMA wire from memory).

DFSDM must be globally disabled (by DFSDMEN=0 in DFSDM_CH0CFGR1) before stopping the system clock to enter in the STOP mode of the device.

DFSDM clocks

The internal DFSDM clock $f_{DFSDMCLK}$, which is used to drive the channel transceivers, digital processing blocks (digital filter, integrator) and next additional blocks (analog watchdog, short-circuit detector, extremes detector, control block) is generated by the RCC block and is derived from the system clock SYSCLK or peripheral clock PCLK2 (see *Section 10.7.49: RCC SAI1 kernel clock selection register (RCC_SAI1CKSELR)*). The DFSDM clock is automatically stopped in stop mode (if DFEN = 0 for all DFSDM_FLTx, x=0..5).

The DFSDM serial channel transceivers can receive an external serial clock to sample an external serial data stream. The internal DFSDM clock must be at least 4 times faster than the external serial clock if standard SPI coding is used, and 6 times faster than the external serial clock if Manchester coding is used.

DFSDM can provide one external output clock signal to drive external ΣΔ modulator(s) clock input(s). It is provided on CKOUT pin. This output clock signal must be in the range specified in given device datasheet and is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) by programmable divider in the range 2 - 256 (CKOUTDIV in DFSDM_CH0CFGR1 register). The audio clock source is the clock selected by RCC configuration (see).

32.4.4 Serial channel transceivers

There are 8 multiplexed serial data channels which can be selected for conversion by each filter or Analog watchdog or Short-circuit detector. Those serial transceivers receive data stream from external ΣΔ modulator. Data stream can be sent in SPI format or Manchester coded format (see SITP[1:0] bits in DFSDM_CHyCFGR1 register).
The channel is enabled for operation by setting CHEN=1 in DFSDM_CHyCFGR1 register.

Channel inputs selection

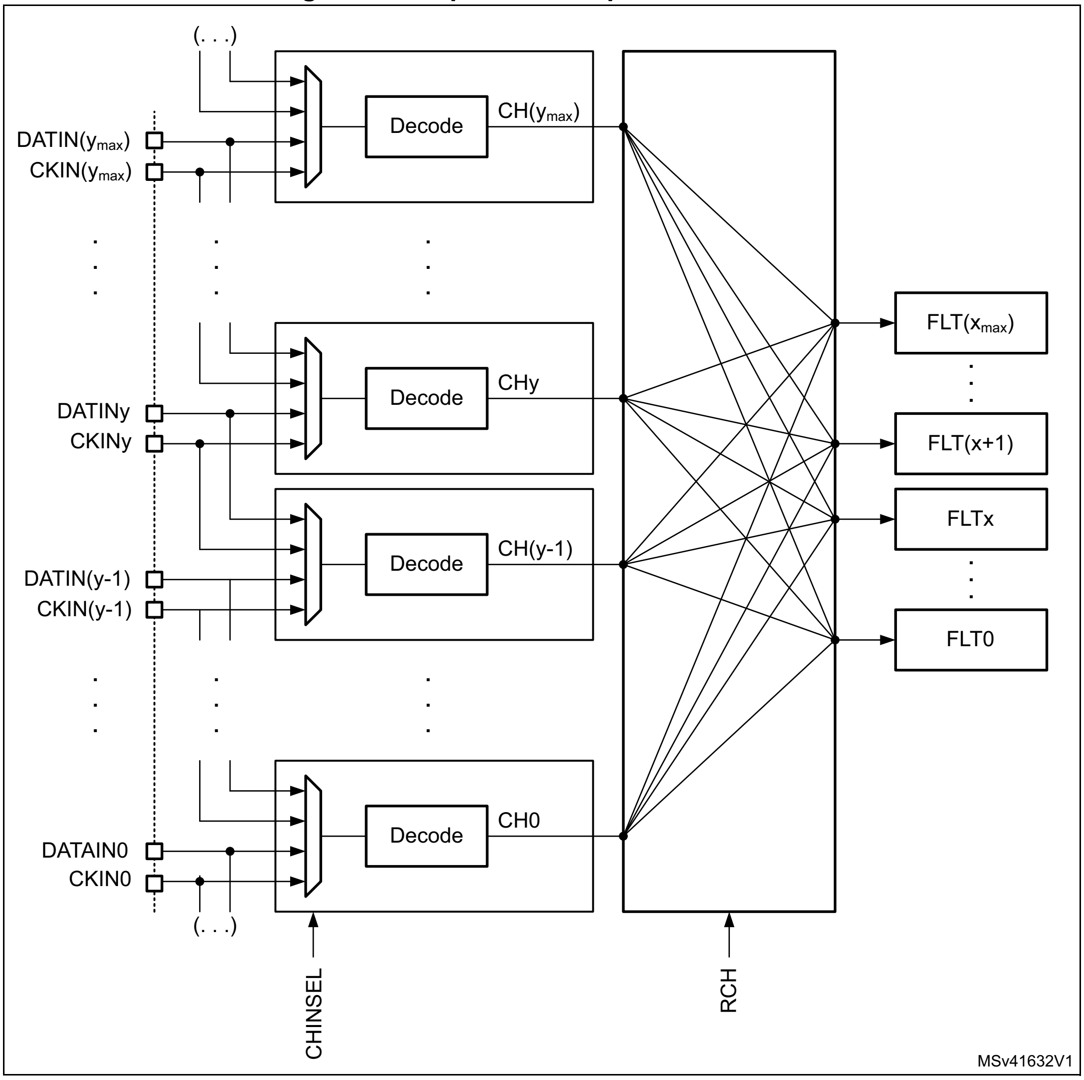
Serial inputs (data and clock signals) from DATINy and CKINy pins can be redirected from the following channel pins. This serial input channel redirection is set by CHINSEL bit in DFSDM_CHyCFGR1 register.

Channel redirection can be used to collect audio data from PDM (pulse density modulation) stereo microphone type. PDM stereo microphone has one data and one clock signal. Data signal provides information for both left and right audio channel (rising clock edge samples for left channel and falling clock edge samples for right channel).

Configuration of serial channels for PDM microphone input:

- PDM microphone signals (data, clock) will be connected to DFSDM input serial channel y (DATINy, CKOUT) pins.
- Channel y will be configured: CHINSEL = 0 (input from given channel pins: DATINy, CKINy).
- Channel (y-1) (modulo 8) will be configured: CHINSEL = 1 (input from the following channel ((y-1)+1) pins: DATINy, CKINy).
- Channel y: SITP[1:0] = 0 (rising edge to strobe data) => left audio channel on channel y.
- Channel (y-1): SITP[1:0] = 1 (falling edge to strobe data) => right audio channel on channel y-1.
- Two DFSDM filters will be assigned to channel y and channel (y-1) (to filter left and right channels from PDM microphone).

Figure 262. Input channel pins redirection



Output clock generation

A clock signal can be provided on CKOUT pin to drive external ΣΔ modulator clock inputs. The frequency of this CKOUT signal is derived from DFSDM clock or from audio clock (see CKOUTSRC bit in DFSDM_CH0CFGR1 register) divided by a predivider (see CKOUTDIV bits in DFSDM_CH0CFGR1 register). If the output clock is stopped, then CKOUT signal is set to low state (output clock can be stopped by CKOUTDIV=0 in DFSDM_CHyCFGR1 register or by DFSDMEN=0 in DFSDM_CH0CFGR1 register). The output clock stopping is performed:

- 4 system clocks after DFSDMEN is cleared (if CKOUTSRC=0)
- 1 system clock and 3 audio clocks after DFSDMEN is cleared (if CKOUTSRC=1)

Before changing CKOUTSRC the software has to wait for CKOUT being stopped to avoid glitch on CKOUT pin. The output clock signal frequency must be in the range 0 - 20 MHz.

SPI data input format operation

In SPI format, the data stream is sent in serial format through data and clock signals. Data signal is always provided from DATINy pin. A clock signal can be provided externally from CKINy pin or internally from a signal derived from the CKOUT signal source.

In case of external clock source selection (SPICKSEL[1:0]=0) data signal (on DATINy pin) is sampled on rising or falling clock edge (of CKINy pin) according SITP[1:0] bits setting (in DFSDM_CHyCFGR1 register).

Internal clock sources - see SPICKSEL[1:0] in DFSDM_CHyCFGR1 register:

- CKOUT signal:
  - For connection to external ΣΔ modulator which uses directly its clock input (from CKOUT) to generate its output serial communication clock.
  - Sampling point: on rising/falling edge according SITP[1:0] setting.
- CKOUT/2 signal (generated on CKOUT rising edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
  - Sampling point: on each second CKOUT falling edge.
- CKOUT/2 signal (generated on CKOUT falling edge):
  - For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
  - Sampling point: on each second CKOUT rising edge.

Note: An internal clock source can only be used when the external ΣΔ modulator uses CKOUT signal as a clock input (to have synchronous clock and data operation).

Internal clock source usage can save CKINy pin connection (CKINy pins can be used for other purpose).

The clock source signal frequency must be in the range 0 - 20 MHz for SPI coding and less than $f_{DFSDMCLK}/4$.

Manchester coded data input format operation

In Manchester coded format, the data stream is sent in serial format through DATINy pin only. Decoded data and clock signal are recovered from serial stream after Manchester decoding. There are two possible settings of Manchester codings (see SITP[1:0] bits in DFSDM_CHyCFGR1 register):

- signal rising edge = log 0; signal falling edge = log 1
- signal rising edge = log 1; signal falling edge = log 0

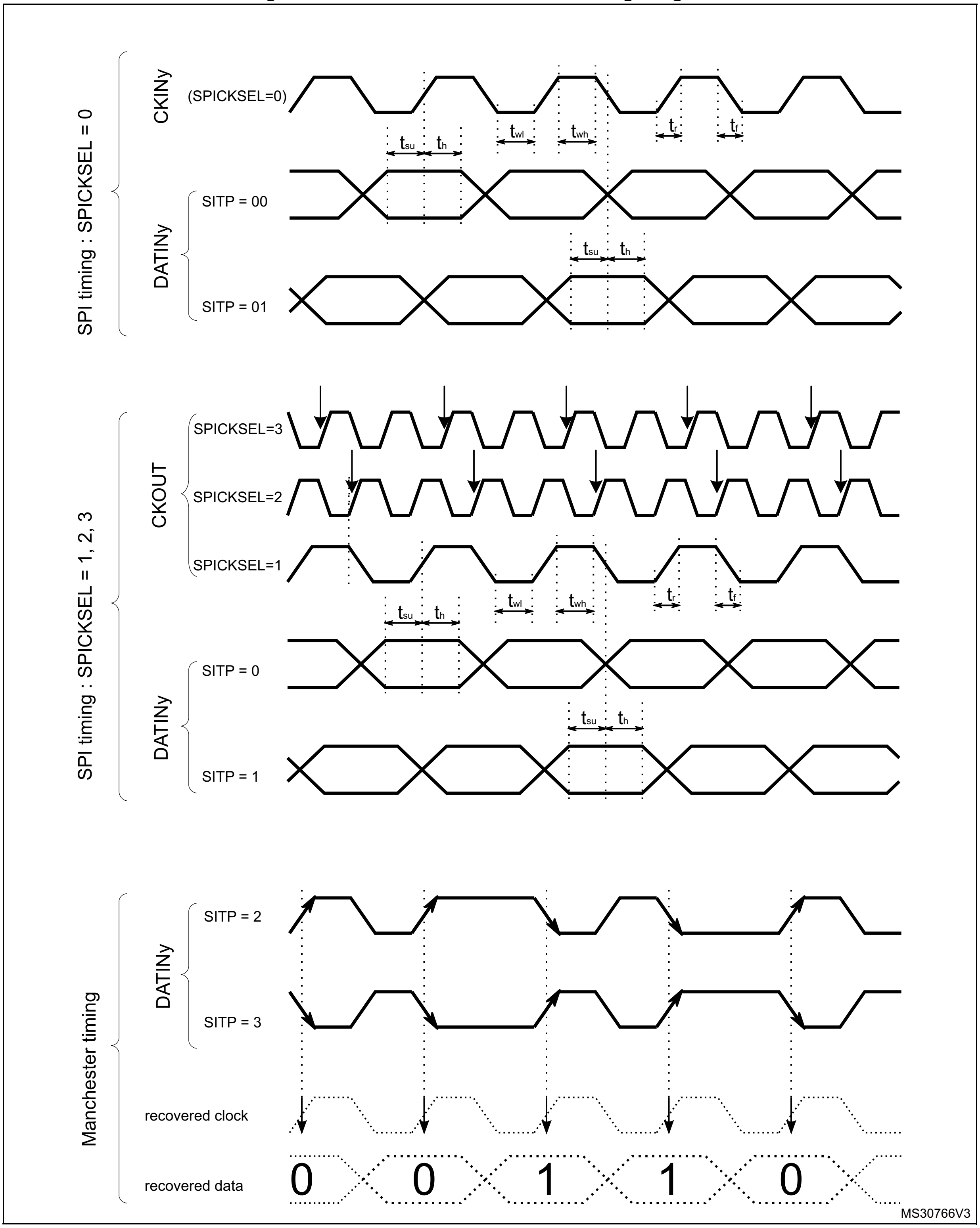
The recovered clock signal frequency for Manchester coding must be in the range 0 - 10 MHz and less than $f_{DFSDMCLK}/6$.

To correctly receive Manchester coded data, the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according formula:

$$((\text{CKOUTDIV} + 1) \times T_{\text{INCKOUT}}) < T_{\text{Manchester clock}} < (2 \times \text{CKOUTDIV} \times T_{\text{INCKOUT}})$$

Where $T_{INCKOUT}$ is the input clock into CKOUT before division by CKOUTDIV divider.

Figure 263. Channel transceiver timing diagrams



Clock absence detection

Channels serial clock inputs can be checked for clock absence/presence to ensure the correct operation of conversion and error reporting. Clock absence detection can be enabled or disabled on each input channel y by bit CKABEN in DFSDM_CHyCFGR1 register. If enabled, then this clock absence detection is performed continuously on a given channel. A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock error (see CKABF[7:0] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed. Clock absence status bit CKABF[y] is set also by hardware when corresponding channel y is disabled (if CHEN[y] = 0 then CKABF[y] is held in set state).

When a clock absence event has occurred, the data conversion (and/or analog watchdog and short-circuit detector) provides incorrect data. The user should manage this event and discard given data while a clock absence is reported.

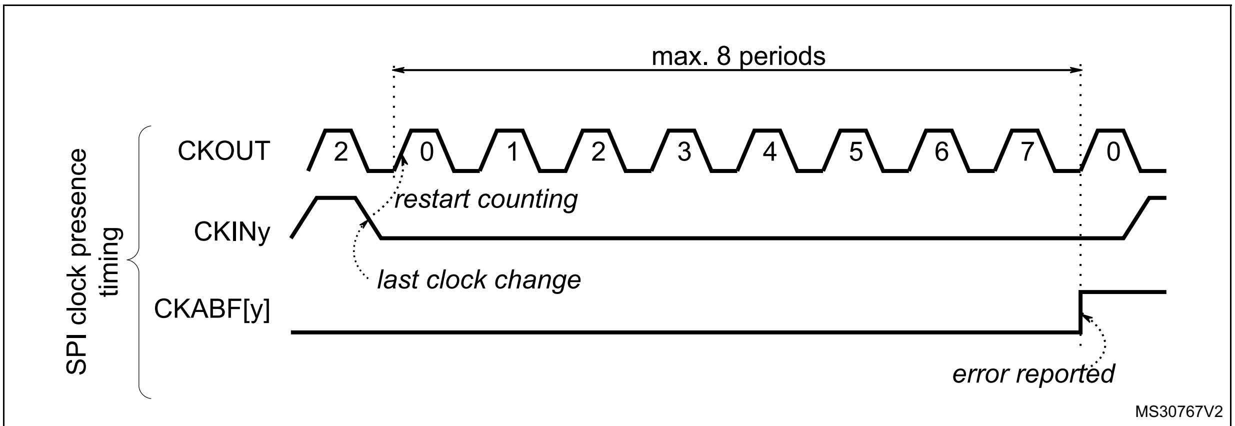
The clock absence feature is available only when the system clock is used for the CKOUT signal (CKOUTSRC=0 in DFSDM_CH0CFGR1 register).

When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register). The software sequence concerning clock absence detection feature should be:

- Enable given channel by CHEN = 1
- Try to clear the clock absence flag (by CLRCKABF = 1) until the clock absence flag is really cleared (CKABF = 0). At this time, the transceiver is synchronized (signal clock is valid) and is able to receive data.
- Enable the clock absence feature CKABEN = 1 and the associated interrupt CKABIE = 1 to detect if the SPI clock is lost or Manchester data edges are missing.

If SPI data format is used, then the clock absence detection is based on the comparison of an external input clock with an output clock generation (CKOUT signal). The external input clock signal into the input channel must be changed at least once per 8 signal periods of CKOUT signal (which is controlled by CKOUTDIV field in DFSDM_CH0CFGR1 register).

Figure 264. Clock absence timing diagram for SPI



If Manchester data format is used, then the clock absence means that the clock recovery is unable to perform from Manchester coded signal. For a correct clock recovery, it is first necessary to receive data with 1 to 0 or 0 to 1 transition (see *Figure 266* for Manchester synchronization).

The detection of a clock absence in Manchester coding (after a first successful synchronization) is based on changes comparison of coded serial data input signal with output clock generation (CKOUT signal). There must be a voltage level change on DATINy pin during 2 periods of CKOUT signal (which is controlled by CKOUTDIV bits in DFSDM_CH0CFGR1 register). This condition also defines the minimum data rate to be able to correctly recover the Manchester coded data and clock signals.

The maximum data rate of Manchester coded data must be less than the CKOUT signal.

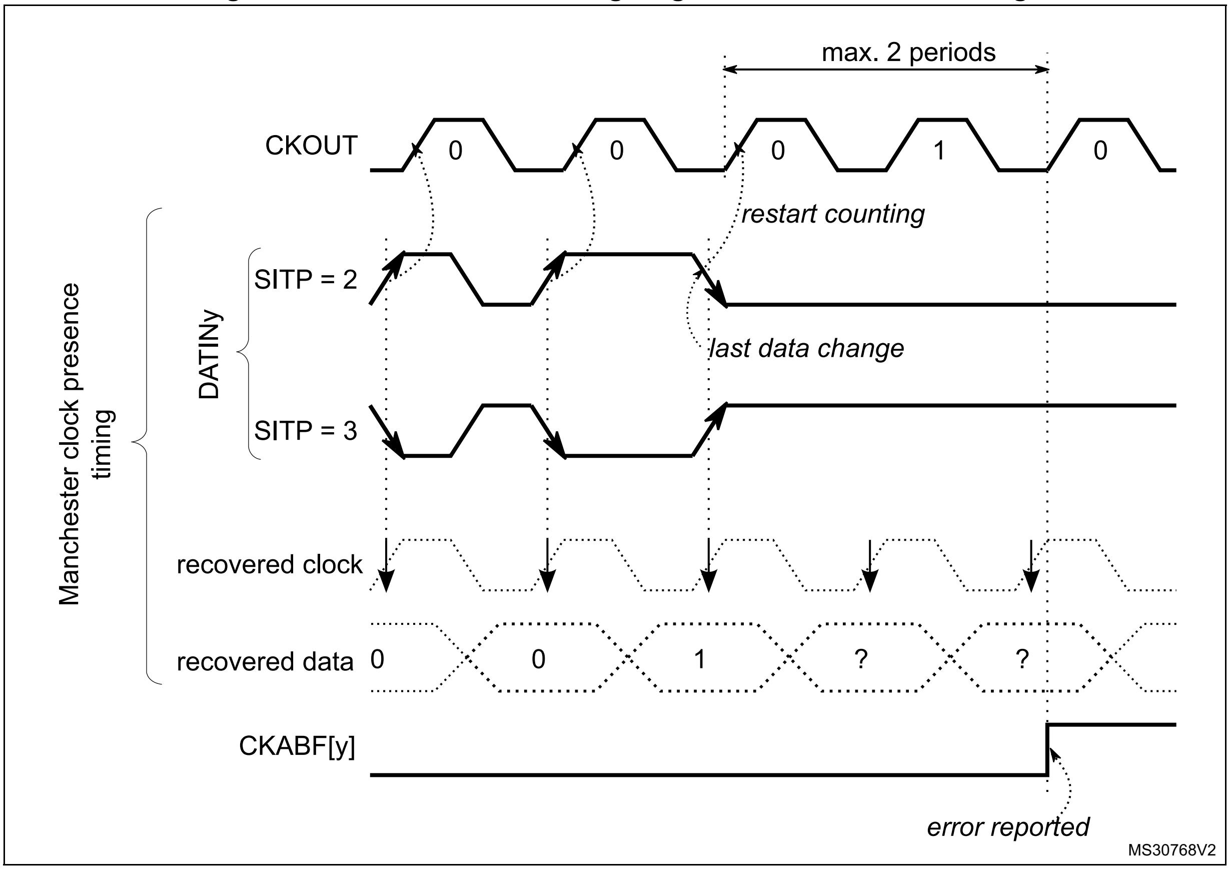
So to correctly receive Manchester coded data, the CKOUTDIV divider must be set according the formula:

$$((\mathrm{CKOUTDIV} + 1) \times T_{\mathrm{INCKOUT}}) < T_{\mathrm{Manchester\ clock}} < (2 \times \mathrm{CKOUTDIV} \times T_{\mathrm{INCKOUT}})$$

Where $T_{INCKOUT}$ is the input clock into CKOUT before division by CKOUTDIV divider.

A clock absence flag is set (CKABF[y] = 1) and an interrupt can be invoked (if CKABIE=1) in case of an input clock recovery error (see CKABF[7:0] in DFSDM_FLT0ISR register and CKABEN in DFSDM_CHyCFGR1). After a clock absence flag clearing (by CLRCKABF in DFSDM_FLT0ICR register), the clock absence flag is refreshed.

Figure 265. Clock absence timing diagram for Manchester coding



Manchester/SPI code synchronization

The Manchester coded stream must be synchronized the first time after enabling the channel (CHEN=1 in DFSDM_CHyCFGR1 register). The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (to be able to detect valid data edge) is received. The end of the synchronization can be checked by polling CKABF[y]=0 for a given channel after it has been cleared by CLRCKABF[y] in DFSDM_FLT0ICR, following the software sequence detailed hereafter:

CKABF[y] flag is cleared by setting CLRCKABF[y] bit. If channel y is not yet synchronized the hardware immediately set the CKABF[y] flag. Software is then reading back the CKABF[y] flag and if it is set then perform again clearing of this flag by setting CLRCKABF[y] bit. This software sequence (polling of CKABF[y] flag) continues until CKABF[y] flag is set (signalizing that Manchester stream is synchronized). To be able to synchronize/receive Manchester coded data the CKOUTDIV divider (in DFSDM_CH0CFGR1 register) must be set with respect to expected Manchester data rate according the formula below.

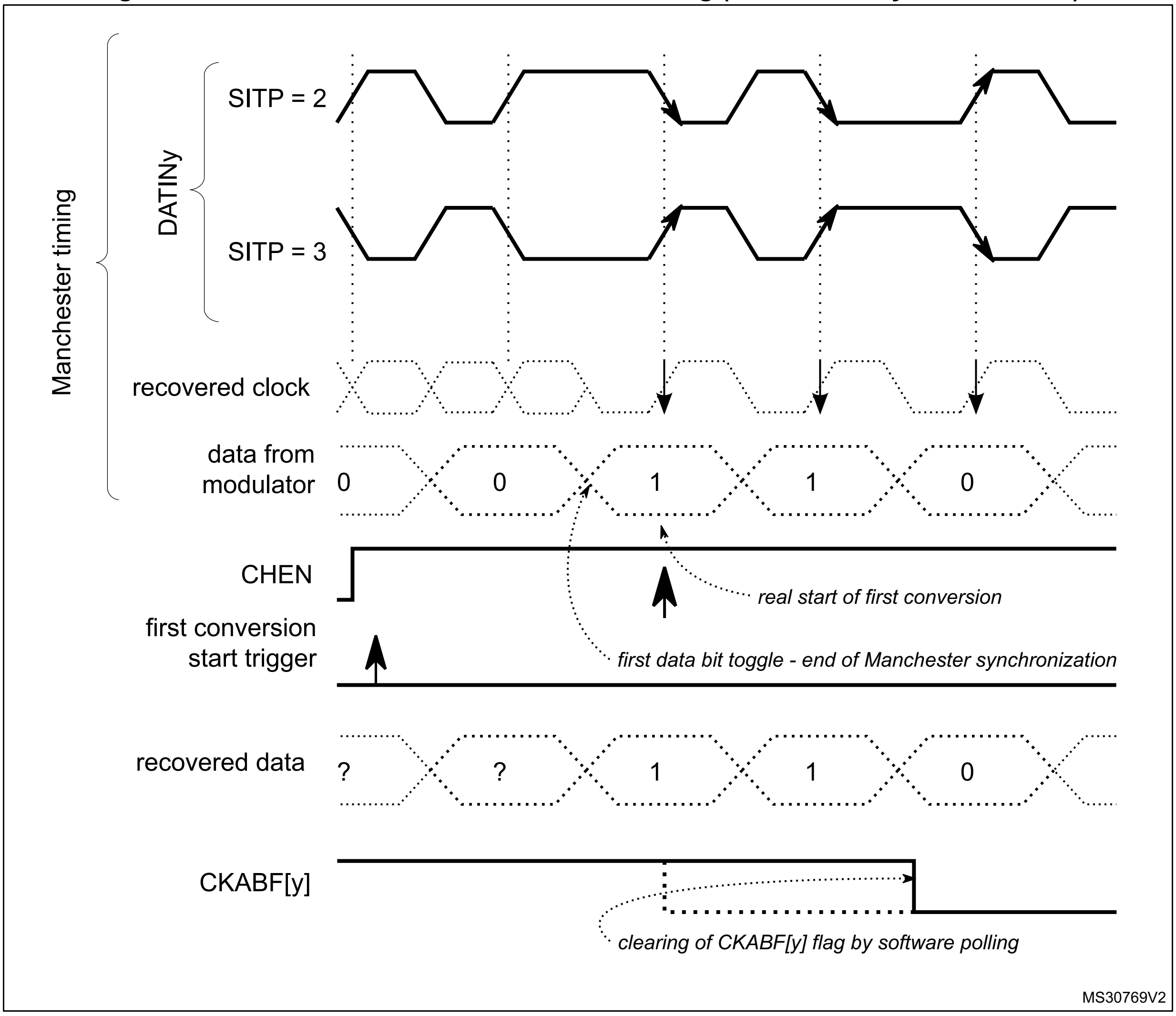
$$((\mathrm{CKOUTDIV} + 1) \times T_{\mathrm{INCKOUT}}) < T_{\mathrm{Manchester\ clock}} < (2 \times \mathrm{CKOUTDIV} \times T_{\mathrm{INCKOUT}})$$

Where $T_{INCKOUT}$ is the input clock into CKOUT before division by CKOUTDIV divider.

SPI coded stream is synchronized after first detection of clock input signal (valid rising/falling edge).

Note: When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y] bit (in DFSDM_FLT0ICR register).

Figure 266. First conversion for Manchester coding (Manchester synchronization)



External serial clock frequency measurement

The measuring of a channel serial clock input frequency provides a real data rate from an external ΣΔ modulator, which is important for application purposes.

An external serial clock input frequency can be measured by a timer counting DFSDM clocks ($f_{DFSDMCLK}$) during one conversion duration. The counting starts at the first input data clock after a conversion trigger (regular or injected) and finishes by last input data clock before conversion ends (end of conversion flag is set). Each conversion duration (time between first serial sample and last serial sample) is updated in counter CNVCNT[27:0] in register DFSDM_FLTxCNVTIMR when the conversion finishes (JEOCF=1 or REOCF=1). The user can then compute the data rate according to the digital filter settings (FORD, FOSR, IOSR, FAST). The external serial frequency measurement is stopped only if the filter is bypassed (FOSR=0, only integrator is active, CNVCNT[27:0]=0 in DFSDM_FLTxCNVTIMR register).

In case of parallel data input (*Section 32.4.6: Parallel data inputs*) the measured frequency is the average input data rate during one conversion.

Note: When conversion is interrupted (e.g. by disabling/enabling the selected channel) the interruption time is also counted in CNVCNT[27:0]. Therefore it is recommended to not interrupt the conversion for correct conversion duration result.

Conversion times:

injected conversion or regular conversion with FAST = 0 (or first conversion if FAST=1):

for $Sinc^x$ filters (x=1..5):

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$

for FastSinc filter:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$

regular conversion with FAST = 1 (except first conversion):

for $Sinc^x$ and FastSinc filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

where:

- f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate (in case of parallel data input)
- F_{OSR} is the filter oversampling ratio: F_{OSR} = FOSR[9:0]+1 (see DFSDM_FLTxFCR register)
- I_{OSR} is the integrator oversampling ratio: I_{OSR} = IOSR[7:0]+1 (see DFSDM_FLTxFCR register)
- F_{ORD} is the filter order: F_{ORD} = FORD[2:0] (see DFSDM_FLTxFCR register)

Channel offset setting

Each channel has its own offset setting (in register) which is finally subtracted from each conversion result (injected or regular) from a given channel. Offset correction is performed after the data right bit shift. The offset is stored as a 24-bit signed value in OFFSET[23:0] field in DFSDM_CHyCFGR2 register.

Data right bit shift

To have the result aligned to a 24-bit value, each channel defines a number of right bit shifts which will be applied on each conversion result (injected or regular) from a given channel. The data bit shift number is stored in DTRBS[4:0] bits in DFSDM_CHyCFGR2 register.

The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained, in order to have valid 24-bit signed format of result data.

Pulses skipper

Purpose of the pulses skipper is to implement delay line like behavior for given input channel(s). Given number of samples from input serial data stream (serial stream only) can be discarded before they enter into the filter. This data discarding is performed by skipping given number of sampling input clock pulses (given serial data samples are then not sampled by filter). The sampling clock is gated by pulses skipper function for given number of clock pulses. When given clock pulses are skipped then the filtering continues for following input data. With comparison to non skipped data stream this operation causes that the final output sample (and next samples) from filter will be calculated from later input data. This final sample then looks a bit in forward - because it is calculated from newer input samples than the “non-skipped” sample. The final “skipped sample” is converted later because the skipped input data samples must be replaced by followed input data samples. The final data buffers behavior (skipped and non-skipped output data buffers comparison) looks like the non-skipped data stream is a bit delayed - both data buffers will be phase shifted.

Number of clock pulses to be skipped should be written into PLSSKP[5:0] field in DFSDM_CHyDLYR register. Once PLSSKP[5:0] field is written the execution of pulses skipping is started on given channel. PLSSKP[5:0] field can be read in order to check the progress of pulses skipper. When PLSSKP[5:0]=0 means that pulses skipping has been executed.

Up to 63 clock pulses can be skip with a single write operation into PLSSKP[5:0]. If more pulses need to be skipped, then user has to write several times into the PLSSKP[5:0] field. The application software should handle cumulative skipped clock number per each filter.

32.4.5 Configuring the input serial interface

The following parameters must be configured for the input serial interface:

- **Output clock predivider**. There is a programmable predivider to generate the output clock from DFSDM clock (2 - 256). It is defined by CKOUTDIV[7:0] bits in DFSDM_CH0CFGR1 register.
- **Serial interface type and input clock phase.** Selection of SPI or Manchester coding and sampling edge of input clock. It is defined by SITP [1:0] bits in DFSDM_CHyCFGR1 register.
- **Input clock source.** External source from CKINy pin or internal from CKOUT pin. It is defined by SPICKSEL[1:0] field in DFSDM_CHyCFGR1 register.
- **Final data right bit-shift.** Defines the final data right bit shift to have the result aligned to a 24-bit value. It is defined by DTRBS[4:0] in DFSDM_CHyCFGR2 register.
- **Channel offset per channel.** Defines the analog offset of a given serial channel (offset of connected external ΣΔ modulator). It is defined by OFFSET[23:0] bits in DFSDM_CHyCFGR2 register.
- **short-circuit detector and clock absence per channel enable.** To enable or disable the short-circuit detector (by SCDEN bit) and the clock absence monitoring (by CKABEN bit) on a given serial channel in register DFSDM_CHyCFGR1.
- **Analog watchdog filter and short-circuit detector threshold settings.** To configure channel analog watchdog filter parameters and channel short-circuit detector parameters. Configurations are defined in DFSDM_CHyAWSCDR register.

32.4.6 Parallel data inputs

Each input channel provides a register for 16-bit parallel data input (besides serial data input). Each 16-bit parallel input can be sourced from internal data sources only:

- internal ADC results
- direct CPU/DMA writing.

The selection for using serial or parallel data input for a given channel is done by field DATMPX[1:0] of DFSDM_CHyCFGR1 register. In DATMPX[1:0] is also defined the parallel data source: internal ADC or direct write by CPU/DMA.

Each channel contains a 32-bit data input register DFSDM_CHyDATINR in which it can be written a 16-bit data. Data are in 16-bit signed format. Those data can be used as input to the digital filter which is accepting 16-bit parallel data.

If serial data input is selected (DATMPX[1:0] = 0), the DFSDM_CHyDATINR register is write protected.

Input from internal ADC

In case of ADC data parallel input (DATMPX[1:0]=1) the ADC[y+1] result is assigned to channel y input (ADC1 is filling DFSDM_CHDATIN0R register, ADC2 is filling DFSDM_CHDATIN1R register, ... , ADC8 is filling DFSDM_CHDATIN7R register). End of conversion event from ADC[y+1] causes update of channel y data (parallel data from ADC[y+1] are put as next sample to digital filter). Data from ADC[y+1] is written into DFSDM_CHyDATINR register (field INDAT0[15:0]) when end of conversion event occurred.

The setting of data packing mode (DATPACK[1:0] in the DFSDM_CHyCFGR1 register) has no effect in case of ADC data input.

Note: Extension of ADC specification: in case the internal ADC is configured in interleaved mode (e.g. ADC1 together with ADC2 - see ADC specification) then each result from ADC1 or from ADC2 will come to the same 16-bit bus - to the bus of ADC1 - which is coming into DFSDM channel 0 (fixed connection). So there will be double input data rate into DFSDM channel 0 (even samples come from ADC1 and odd samples from ADC2). Channel 1 associated with ADC2 will be free.

Input from memory (direct CPU/DMA write)

The direct data write into DFSDM_CHyDATINR register by CPU or DMA (DATMPX[1:0]=2) can be used as data input in order to process digital data streams from memory or peripherals.

Data can be written by CPU or DMA into DFSDM_CHyDATINR register:

1. **CPU data write:**

   Input data are written directly by CPU into DFSDM_CHyDATINR register.

2. **DMA data write:**

   The DMA should be configured in memory-to-memory transfer mode to transfer data from memory buffer into DFSDM_CHyDATINR register. The destination memory address is the address of DFSDM_CHyDATINR register. Data are transferred at DMA transfer speed from memory to DFSDM parallel input.

   This DMA transfer is different from DMA used to read DFSDM conversion results. Both DMA can be used at the same time - first DMA (configured as memory-to-memory transfer) for input data writings and second DMA (configured as peripheral-to-memory transfer) for data results reading.

The accesses to DFSDM_CHyDATINR can be either 16-bit or 32-bit wide, allowing to load respectively one or two samples in one write operation. 32-bit input data register (DFSDM_CHyDATINR) can be filled with one or two 16-bit data samples, depending on the data packing operation mode defined in field DATPACK[1:0] of DFSDM_CHyCFGR1 register:

1. **Standard mode (DATPACK[1:0]=0):**

   Only one sample is stored in field INDAT0[15:0] of DFSDM_CHyDATINR register which is used as input data for channel y. The upper 16 bits (INDAT1[15:0]) are ignored and write protected. The digital filter must perform one input sampling (from INDAT0[15:0]) to empty data register after it has been filled by CPU/DMA. This mode is used together with 16-bit CPU/DMA access to DFSDM_CHyDATINR register to load one sample per write operation.

2. **Interleaved mode (DATPACK[1:0]=1):**

   DFSDM_CHyDATINR register is used as a two sample buffer. The first sample is stored in INDAT0[15:0] and the second sample is stored in INDAT1[15:0]. The digital filter must perform two input samplings from channel y to empty DFSDM_CHyDATINR register. This mode is used together with 32-bit CPU/DMA access to DFSDM_CHyDATINR register to load two samples per write operation.

3. **Dual mode (DATPACK[1:0]=2):**

   Two samples are written into DFSDM_CHyDATINR register. The data INDAT0[15:0] is for channel y, the data in INDAT1[15:0] is for channel y+1. The data in INDAT1[15:0] is automatically copied INDAT0[15:0] of the following (y+1) channel data register DFSDM_CH[y+1]DATINR). The digital filters must perform two samplings - one from channel y and one from channel (y+1) - in order to empty DFSDM_CHyDATINR registers.

   Dual mode setting (DATPACK[1:0]=2) is available only on even channel numbers (y = 0, 2, 4, 6). If odd channel (y = 1, 3, 5, 7) is set to Dual mode then both INDAT0[15:0] and INDAT1[15:0] parts are write protected for this channel. If even channel is set to Dual mode then the following odd channel must be set into Standard mode (DATPACK[1:0]=0) for correct cooperation with even channels.

See *Figure 267* for DFSDM_CHyDATINR registers data modes and assignments of data samples to channels.

Figure 267. DFSDM_CHyDATINR registers operation modes and assignment

Standard mode 31–16	Standard mode 15–0	Interleaved mode 31–16	Interleaved mode 15–0	Dual mode 31–16	Dual mode 15–0	
Unused	Ch0 (sample 0)	Ch0 (sample 1)	Ch0 (sample 0)	Ch1 (sample 0)	Ch0 (sample 0)	**y = 0**
Unused	Ch1 (sample 0)	Ch1 (sample 1)	Ch1 (sample 0)	Unused	Ch1 (sample 0)	**y = 1**
Unused	Ch2 (sample 0)	Ch2 (sample 1)	Ch2 (sample 0)	Ch3 (sample 0)	Ch2 (sample 0)	**y = 2**
Unused	Ch3 (sample 0)	Ch3 (sample 1)	Ch3 (sample 0)	Unused	Ch3 (sample 0)	**y = 3**
Unused	Ch4 (sample 0)	Ch4 (sample 1)	Ch4 (sample 0)	Ch5 (sample 0)	Ch4 (sample 0)	**y = 4**
Unused	Ch5 (sample 0)	Ch5 (sample 1)	Ch5 (sample 0)	Unused	Ch5 (sample 0)	**y = 5**
Unused	Ch6 (sample 0)	Ch6 (sample 1)	Ch6 (sample 0)	Ch7 (sample 0)	Ch6 (sample 0)	**y = 6**
Unused	Ch7 (sample 0)	Ch7 (sample 1)	Ch7 (sample 0)	Unused	Ch7 (sample 0)	**y = 7**

MS35354V3

The write into DFSDM_CHyDATINR register to load one or two samples must be performed after the selected input channel (channel y) is enabled for data collection (starting conversion for channel y). Otherwise written data are lost for next processing.

For example: for single conversion and interleaved mode, do not start writing pair of data samples into DFSDM_CHyDATINR before the single conversion is started (any data present in the DFSDM_CHyDATINR before starting a conversion is discarded).

32.4.7 Channel selection

There are 8 multiplexed channels which can be selected for conversion using the injected channel group and/or using the regular channel.

The **injected channel group** is a selection of any or all of the 8 channels. JCHG[7:0] in the DFSDM_FLTxJCHGR register selects the channels of the injected group, where JCHG[y]=1 means that channel y is selected.

Injected conversions can operate in scan mode (JSCAN=1) or single mode (JSCAN=0). In scan mode, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first, followed immediately by the next higher channel until all the channels selected by JCHG[7:0] have been converted. In single mode (JSCAN=0), only one channel from the selected channels is converted, and the channel selection is moved to the next channel. Writing to JCHG[7:0] if JSCAN=0 resets the channel selection to the lowest selected channel.

Injected conversions can be launched by software or by a trigger. They are never interrupted by regular conversions.

The **regular channel** is a selection of just one of the 8 channels. RCH[2:0] in the DFSDM_FLTxCR1 register indicates the selected channel.

Regular conversions can be launched only by software (not by a trigger). A sequence of continuous regular conversions is temporarily interrupted when an injected conversion is requested.

Performing a conversion on a disabled channel (CHEN=0 in DFSDM_CHyCFGR1 register) causes that the conversion will never end - because no input data is provided (with no clock signal). In this case, it is necessary to enable a given channel (CHEN=1 in DFSDM_CHyCFGR1 register) or to stop the conversion by DFEN=0 in DFSDM_FLTxCR1 register.

32.4.8 Digital filter configuration

DFSDM contains a $Sinc^x$ type digital filter implementation. This $Sinc^x$ filter performs an input digital data stream filtering, which results in decreasing the output data rate (decimation) and increasing the output data resolution. The $Sinc^x$ digital filter is configurable in order to reach the required output data rates and required output data resolution. The configurable parameters are:

- Filter order/type: (see FORD[2:0] bits in DFSDM_FLTxFCR register):
  - FastSinc
  - $Sinc^1$
  - $Sinc^2$
  - $Sinc^3$
  - $Sinc^4$
  - $Sinc^5$
- Filter oversampling/decimation ratio (see FOSR[9:0] bits in DFSDM_FLTxFCR register):
  - FOSR = 1-1024 - for FastSinc filter and $Sinc^x$ filter x = F_{ORD} = 1..3
  - FOSR = 1-215 - for $Sinc^x$ filter x = F_{ORD} = 4
  - FOSR = 1-73 - for $Sinc^x$ filter x = F_{ORD} = 5

The filter has the following transfer function (impulse response in H domain):

- Sinc^x filter type: $H(z) = \left(\dfrac{1-z^{-FOSR}}{1-z^{-1}}\right)^{x}$

- FastSinc filter type: $H(z) = \left(\dfrac{1-z^{-FOSR}}{1-z^{-1}}\right)^{2} \cdot (1+z^{-(2 \cdot FOSR)})$

Figure 268. Example: Sinc^3 filter response

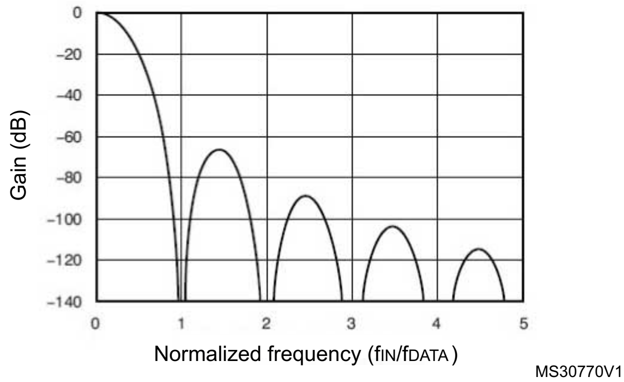


Table 225. Filter maximum output resolution (peak data values from filter output) for some FOSR values

FOSR	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
x	+/- x	+/- x^2	+/- $2x^2$	+/- x^3	+/- x^4	+/- x^5
4	+/- 4	+/- 16	+/- 32	+/- 64	+/- 256	+/- 1024
8	+/- 8	+/- 64	+/- 128	+/- 512	+/- 4096	-
32	+/- 32	+/- 1024	+/- 2048	+/- 32768	+/- 1048576	+/- 33554432
64	+/- 64	+/- 4096	+/- 8192	+/- 262144	+/- 16777216	+/- 1073741824
128	+/- 128	+/- 16384	+/- 32768	+/- 2097152	+/- 268435456	
256	+/- 256	+/- 65536	+/- 131072	+/- 16777216	Result can overflow on full scale input (> 32-bit signed integer)	
1024	+/- 1024	+/- 1048576	+/- 2097152	+/- 1073741824		

For more information about Sinc filter type properties and usage, it is recommended to study the theory about digital filters (more resources can be downloaded from internet).

32.4.9 Integrator unit

The integrator performs additional decimation and a resolution increase of data coming from the digital filter. The integrator simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The integrator oversampling ratio parameter defines how many data counts will be summed to one data output from the integrator. IOSR can be set in the range 1-256 (see IOSR[7:0] bits description in DFSDM_FLTxFCR register).

Table 226. Integrator maximum output resolution (peak data values from integrator output) for some IOSR values and FOSR = 256 and $Sinc^3$ filter type (largest data)

IOSR	$Sinc^1$	$Sinc^2$	FastSinc	$Sinc^3$	$Sinc^4$	$Sinc^5$
x	+/- FOSR. x	+/- $FOSR^2$. x	+/- 2.$FOSR^2$. x	+/- $FOSR^3$. x	+/- $FOSR^4$. x	+/- $FOSR^5$. x
4	-	-	-	+/- 67 108 864	-	-
32	-	-	-	+/- 536 870 912	-	-
128	-	-	-	+/- 2 147 483 648	-	-
256	-	-	-	+/- 2^{32}	-	-

32.4.10 Analog watchdog

The analog watchdog purpose is to trigger an external signal (break or interrupt) when an analog signal reaches or crosses given maximum and minimum threshold values. An interrupt/event/break generation can then be invoked.

Each analog watchdog will supervise serial data receiver outputs (after the analog watchdog filter on each channel) or data output register (current injected or regular conversion result) according to AWFSEL bit setting (in DFSDM_FLTxCR1 register). The input channels to be monitored or not by the analog watchdog x will be selected by AWDCH[7:0] in DFSDM_FLTxCR2 register.

Analog watchdog conversions on input channels are independent from standard conversions. In this case, the analog watchdog uses its own filters and signal processing on each input channel independently from the main injected or regular conversions. Analog watchdog conversions are performed in a continuous mode on the selected input channels in order to watch channels also when main injected or regular conversions are paused (RCIP = 0, JCIP = 0).

There are high and low threshold registers which are compared with given data values (set by AWHT[23:0] bits in DFSDM_FLTxAWHTR register and by AWLT[23:0] bits in DFSDM_FLTxAWLTR register).

There are 2 options for comparing the threshold registers with the data values

- Option1: in this case, the input data are taken from final output data register (AWFSEL=0). This option is characterized by:
  - high input data resolution (up to 24-bits)
  - slow response time - inappropriate for fast response applications like overcurrent detection
  - for the comparison the final data are taken after bit shifting and offset data correction
  - final data are available only after main regular or injected conversions are performed
  - can be used in case of parallel input data source (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register)
- Option2: in this case, the input data are taken from any serial data receivers output (AWFSEL=1). This option is characterized by:
  - input serial data are processed by dedicated analog watchdog $Sinc^x$ channel filters with configurable oversampling ratio (1..32) and filter order (1..3) (see AWFOSR[4:0] and AWFORD[1:0] bits setting in DFSDM_CHyAWSCDR register)
  - lower resolution (up to 16-bit)
  - fast response time - appropriate for applications which require a fast response like overcurrent/overvoltage detection)
  - data are available in continuous mode independently from main regular or injected conversions activity

In case of input channels monitoring (AWFSEL=1), the data for comparison to threshold is taken from channels selected by AWDCH[7:0] field (DFSDM_FLTxCR2 register). Each of the selected channels filter result is compared to one threshold value pair (AWHT[23:0] / AWLT[23:0]). In this case, only higher 16 bits (AWHT[23:8] / AWLT[23:8]) define the 16-bit threshold compared with the analog watchdog filter output because data coming from the analog watchdog filter is up to a 16-bit resolution. Bits AWHT[7:0] / AWLT[7:0] are not taken into comparison in this case (AWFSEL=1).

Parameters of the analog watchdog filter configuration for each input channel are set in DFSDM_CHyAWSCDR register (filter order AWFORD[1:0] and filter oversampling ratio AWFOSR[4:0]).

Each input channel has its own comparator which compares the analog watchdog data (from analog watchdog filter) with analog watchdog threshold values (AWHT/AWLT). When several channels are selected (field AWDCH[7:0] field of DFSDM_FLTxCR2 register), several comparison requests may be received simultaneously. In this case, the channel request with the lowest number is managed first and then continuing to higher selected channels. For each channel, the result can be recorded in a separate flag (fields AWHTF[7:0], AWLTF[7:0] of DFSDM_FLTxAWSR register). Each channel request is executed in 8 DFSDM clock cycles. So, the bandwidth from each channel is limited to 8 DFSDM clock cycles (if AWDCH[7:0] = 0xFF). Because the maximum input channel sampling clock frequency is the DFSDM clock frequency divided by 4, the configuration AWFOSR = 0 (analog watchdog filter is bypassed) cannot be used for analog watchdog feature at this input clock speed. Therefore user must properly configure the number of watched channels and analog watchdog filter parameters with respect to input sampling clock speed and DFSDM frequency.

Analog watchdog filter data for given channel y is available for reading by firmware on field WDATA[15:0] in DFSDM_CHyWDATR register. That analog watchdog filter data is converted continuously (if CHEN=1 in DFSDM_CHyCFGR1 register) with the data rate given by the analog watchdog filter setting and the channel input clock frequency.

The analog watchdog filter conversion works like a regular Fast Continuous Conversion without the intergator. The number of serial samples needed for one result from analog watchdog filter output (at channel input clock frequency f_{CKIN}):

first conversion:

for $Sinc^x$ filters (x=1..5): number of samples = [F_{OSR} * F_{ORD} + F_{ORD} + 1]

for FastSinc filter: number of samples = [F_{OSR} * 4 + 2 + 1]

next conversions:

for $Sinc^x$ and FastSinc filters: number of samples = [FOSR * IOSR]

where:

F_{OSR} .... filter oversampling ratio: F_{OSR} = AWFOSR[4:0]+1 (see DFSDM_CHyAWSCDR register)

F_{ORD} .... the filter order: F_{ORD} = AWFORD[1:0] (see DFSDM_CHyAWSCDR register)

In case of output data register monitoring (AWFSEL=0), the comparison is done after a right bit shift and an offset correction of final data (see OFFSET[23:0] and DTRBS[4:0] fields in DFSDM_CHyCFGR2 register). A comparison is performed after each injected or regular end of conversion for the channels selected by AWDCH[7:0] field (in DFSDM_FLTxCR2 register).

The status of an analog watchdog event is signalized in DFSDM_FLTxAWSR register where a given event is latched. AWHTF[y]=1 flag signalizes crossing AWHT[23:0] value on channel y. AWLTF[y]=1 flag signalizes crossing AWLT[23:0] value on channel y. Latched events in DFSDM_FLTxAWSR register are cleared by writing ‘1’ into the corresponding clearing bit CLRAWHTF[y] or CLRAWLTF[y] in DFSDM_FLTxAWCFR register.

The global status of an analog watchdog is signalized by the AWDF flag bit in DFSDM_FLTxISR register (it is used for the fast detection of an interrupt source). AWDF=1 signalizes that at least one watchdog occurred (AWHTF[y]=1 or AWLTF[y]=1 for at least one channel). AWDF bit is cleared when all AWHTF[7:0] and AWLTF[7:0] are cleared.

An analog watchdog event can be assigned to break output signal. There are four break outputs to be assigned to a high or low threshold crossing event (dfsdm_break[3:0]). The break signal assignment to a given analog watchdog event is done by BKAWH[3:0] and BKAWL[3:0] fields in DFSDM_FLTxAWHTR and DFSDM_FLTxAWLTR register.

32.4.11 Short-circuit detector

The purpose of a short-circuit detector is to signalize with a very fast response time if an analog signal reached saturated values (out of full scale ranges) and remained on this value given time. This behavior can detect short-circuit or open circuit errors (e.g. overcurrent or overvoltage). An interrupt/event/break generation can be invoked.

Input data into a short-circuit detector is taken from channel transceiver outputs.

There is an upcounting counter on each input channel which is counting consecutive 0’s or 1’s on serial data receiver outputs. A counter is restarted if there is a change in the data stream received - 1 to 0 or 0 to 1 change of data signal. If this counter reaches a short-circuit threshold register value (SCDT[7:0] bits in DFSDM_CHyAWSCDR register), then a short-circuit event is invoked. Each input channel has its short-circuit detector. Any channel can be selected to be continuously monitored by setting the SCDEN bit (in DFSDM_CHyCFGR1 register) and it has its own short-circuit detector settings (threshold value in SCDT[7:0] bits, status bit SCDF[7:0], status clearing bits CLRSCDF[7:0]). Status flag SCDF[y] is cleared also by hardware when corresponding channel y is disabled (CHEN[y] = 0).

On each channel, a short-circuit detector event can be assigned to break output signal dfsdm_break[3:0]. There are four break outputs to be assigned to a short-circuit detector event. The break signal assignment to a given channel short-circuit detector event is done by BKSCD[3:0] field in DFSDM_CHyAWSCDR register.

Short circuit detector cannot be used in case of parallel input data channel selection (DATMPX[1:0] ≠ 0 in DFSDM_CHyCFGR1 register).

Four break outputs are totally available (shared with the analog watchdog function).

32.4.12 Extreme detector

The purpose of an extremes detector is to collect the minimum and maximum values of final output data words (peak to peak values).

If the output data word is higher than the value stored in the extremes detector maximum register (EXMAX[23:0] bits in DFSDM_FLTxEXMAX register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMAXCH[2:0] bits (in DFSDM_FLTxEXMAX register) .

If the output data word is lower than the value stored in the extremes detector minimum register (EXMIN[23:0] bits in DFSDM_FLTxEXMIN register), then this register is updated with the current output data word value and the channel from which the data is stored is in EXMINCH[2:0] bits (in DFSDM_FLTxEXMIN register).

The minimum and maximum register values can be refreshed by software (by reading given DFSDM_FLTxEXMAX or DFSDM_FLTxEXMIN register). After refresh, the extremes detector minimum data register DFSDM_FLTxEXMIN is filled with 0x7FFFFF (maximum positive value) and the extremes detector maximum register DFSDM_FLTxEXMAX is filled with 0x800000 (minimum negative value).

The extremes detector performs a comparison after a right bit shift and an offset data correction. For each extremes detector, the input channels to be considered into computing the extremes value are selected in EXCH[7:0] bits (in DFSDM_FLTxCR2 register).

32.4.13 Data unit block

The data unit block is the last block of the whole processing path: External ΣΔ modulators - Serial transceivers - Sinc filter - Integrator - Data unit block.

The output data rate depends on the serial data stream rate, and filter and integrator settings. The maximum output data rate is:

$$\mathrm{Datarate}\lceil \mathrm{samples} / \mathrm{s} \rceil = \frac{f_{CKIN}}{F_{OSR} \cdot (I_{OSR} - 1 + F_{ORD}) + (F_{ORD} + 1)} \quad \text{...FAST = 0, Sincx filter}$$

$$\mathrm{Datarate}\lceil \mathrm{samples} / \mathrm{s} \rceil = \frac{f_{CKIN}}{F_{OSR} \cdot (I_{OSR} - 1 + 4) + (2 + 1)} \quad \text{...FAST = 0, FastSinc filter}$$

or

$$\mathrm{Datarate}[\mathrm{samples}\,/\,s] = \frac{f_{CKIN}}{F_{OSR} \cdot I_{OSR}} \qquad \text{...FAST = 1}$$

Maximum output data rate in case of parallel data input:

$$\mathrm{Datarate}[\mathrm{samples}\,/\,s] = \frac{f_{DATAIN\_RATE}}{F_{OSR} \cdot (I_{OSR} - 1 + F_{ORD}) + (F_{ORD} + 1)} \qquad \text{...FAST = 0, Sincx filter}$$

or

$$\mathrm{Datarate}[\mathrm{samples}\,/\,s] = \frac{f_{DATAIN\_RATE}}{F_{OSR} \cdot (I_{OSR} - 1 + 4) + (2 + 1)} \qquad \text{...FAST = 0, FastSinc filter}$$

or

$$\mathrm{Datarate}[\mathrm{samples}\,/\,s] = \frac{f_{DATAIN\_RATE}}{F_{OSR} \cdot I_{OSR}} \qquad \text{...FAST=1 or any filter bypass case } (F_{OSR} = 1)$$

where: f_{DATAIN_RATE}...input data rate from ADC or from CPU/DMA

The right bit-shift of final data is performed in this module because the final data width is 24-bit and data coming from the processing path can be up to 32 bits. This right bit-shift is configurable in the range 0-31 bits for each selected input channel (see DTRBS[4:0] bits in DFSDM_CHyCFGR2 register). The right bit-shift is rounding the result to nearest integer value. The sign of shifted result is maintained - to have valid 24-bit signed format of result data.

In the next step, an offset correction of the result is performed. The offset correction value (OFFSET[23:0] stored in register DFSDM_CHyCFGR2) is subtracted from the output data for a given channel. Data in the OFFSET[23:0] field is set by software by the appropriate calibration routine.

Due to the fact that all operations in digital processing are performed on 32-bit signed registers, the following conditions must be fulfilled not to overflow the result:

$FOSR^{FORD} . IOSR <= 2^{31}$ ... for Sinc^x filters, x = 1..5)
$2 . FOSR^{2} . IOSR <= 2^{31}$ ... for FastSinc filter)

Note: In case of filter and integrator bypass (IOSR[7:0]=0, FOSR[9:0]=0), the input data rate (f_{DATAIN_RATE}) must be limited to be able to read all output data:
$f_{DATAIN\_RATE} \le f_{APB}$
where f_{APB} is the bus frequency to which the DFSDM peripheral is connected.

32.4.14 Signed data format

Each DFSDM input serial channel can be connected to one external ΣΔ modulator. An external ΣΔ modulator can have 2 differential inputs (positive and negative) which can be used for a differential or single-ended signal measurement.

A ΣΔ modulator output is always assumed in a signed format (a data stream of zeros and ones from a ΣΔ modulator represents values -1 and +1).

Signed data format in registers: Data is in a signed format in registers for final output data, analog watchdog, extremes detector, offset correction. The msb of output data word represents the sign of value (two's complement format).

32.4.15 Launching conversions

Injected conversions can be launched using the following methods:

- Software: writing '1' to JSWSTART in the DFSDM_FLTxCR1 register.
- Trigger: JEXTSEL[4:0] selects the trigger signal while JEXTEN activates the trigger and selects the active edge at the same time (see the DFSDM_FLTxCR1 register).
- Synchronous with DFSDM_FLT0 if JSYNC=1: for DFSDM_FLTx (x>0), an injected conversion is automatically launched when in DFSDM_FLT0; the injected conversion is started by software (JSWSTART=1 in DFSDM_FLT0CR2 register). Each injected conversion in DFSDM_FLTx (x>0) is always executed according to its local configuration settings (JSCAN, JCHG, etc.).

If the scan conversion is enabled (bit JSCAN=1) then, each time an injected conversion is triggered, all of the selected channels in the injected group (JCHG[7:0] bits in DFSDM_FLTxJCHGR register) are converted sequentially, starting with the lowest channel (channel 0, if selected).

If the scan conversion is disabled (bit JSCAN=0) then, each time an injected conversion is triggered, only one of the selected channels in the injected group (JCHG[7:0] bits in DFSDM_FLTxJCHGR register) is converted and the channel selection is then moved to the next selected channel. Writing to the JCHG[7:0] bits when JSCAN=0 sets the channel selection to the lowest selected injected channel.

Only one injected conversion can be ongoing at a given time. Thus, any request to launch an injected conversion is ignored if another request for an injected conversion has already been issued but not yet completed.

Regular conversions can be launched using the following methods:

- Software: by writing '1' to RSWSTART in the DFSDM_FLTxCR1 register.
- Synchronous with DFSDM_FLT0 if RSYNC=1: for DFSDM_FLTx (x>0), a regular conversion is automatically launched when in DFSDM_FLT0; a regular conversion is started by software (RSWSTART=1 in DFSDM_FLT0CR2 register). Each regular conversion in DFSDM_FLTx (x>0) is always executed according to its local configuration settings (RCONT, RCH, etc.).

Only one regular conversion can be pending or ongoing at a given time. Thus, any request to launch a regular conversion is ignored if another request for a regular conversion has already been issued but not yet completed. A regular conversion can be pending if it was interrupted by an injected conversion or if it was started while an injected conversion was in progress. This pending regular conversion is then delayed and is performed when all injected conversion are finished. Any delayed regular conversion is signalized by RPEND bit in DFSDM_FLTxRDATAR register.

32.4.16 Continuous and fast continuous modes

Setting RCONT in the DFSDM_FLTxCR1 register causes regular conversions to execute in continuous mode. RCONT=1 means that the channel selected by RCH[2:0] is converted repeatedly after '1' is written to RSWSTART.

The regular conversions executing in continuous mode can be stopped by writing ‘0’ to RCONT. After clearing RCONT, the on-going conversion is stopped immediately.

In continuous mode, the data rate can be increased by setting the FAST bit in the DFSDM_FLTxCR1 register. In this case, the filter does not need to be refilled by new fresh data if converting continuously from one channel because data inside the filter is valid from previously sampled continuous data. The speed increase depends on the chosen filter order. The first conversion in fast mode (FAST=1) after starting a continuous conversion by RSWSTART=1 takes still full time (as when FAST=0), then each subsequent conversion is finished in shorter intervals.

Conversion time in continuous mode:

if FAST = 0 (or first conversion if FAST=1):

for $Sinc^x$ filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$

for FastSinc filter:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$

if FAST = 1 (except first conversion):

for $Sinc^x$ and FastSinc filters:

$t = CNVCNT/f_{DFSDMCLK} = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, only integrator active):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

Continuous mode is not available for injected conversions. Injected conversions can be started by timer trigger to emulate the continuous mode with precise timing.

If a regular continuous conversion is in progress (RCONT=1) and if a write access to DFSDM_FLTxCR1 register requesting regular continuous conversion (RCONT=1) is performed, then regular continuous conversion is restarted from the next conversion cycle (like new regular continuous conversion is applied for new channel selection - even if there is no change in DFSDM_FLTxCR1 register).

32.4.17 Request precedence

An injected conversion has a higher precedence than a regular conversion. A regular conversion which is already in progress is immediately interrupted by the request of an injected conversion; this regular conversion is restarted after the injected conversion finishes.

An injected conversion cannot be launched if another injected conversion is pending or already in progress: any request to launch an injected conversion (either by JSWSTART or by a trigger) is ignored as long as bit JCIP is ‘1’ (in the DFSDM_FLTxISR register).

Similarly, a regular conversion cannot be launched if another regular conversion is pending or already in progress: any request to launch a regular conversion (using RSWSTART) is ignored as long as bit RCIP is ‘1’ (in the DFSDM_FLTxISR register).

However, if an injected conversion is requested while a regular conversion is already in progress, the regular conversion is immediately stopped and an injected conversion is launched. The regular conversion is then restarted and this delayed restart is signalized in bit RPEND.

Injected conversions have precedence over regular conversions in that a injected conversion can temporarily interrupt a sequence of continuous regular conversions. When the sequence of injected conversions finishes, the continuous regular conversions start again if RCONT is still set (and RPEND bit will signalize the delayed start on the first regular conversion result).

Precedence also matters when actions are initiated by the same write to DFSDM, or if multiple actions are pending at the end of another action. For example, suppose that, while an injected conversion is in process (JCIP=1), a single write operation to DFSDM_FLTxCR1 writes ‘1’ to RSWSTART, requesting a regular conversion. When the injected sequence finishes, the precedence dictates that the regular conversion is performed next and its delayed start is signalized in RPEND bit.

32.4.18 Power optimization in run mode

In order to reduce the consumption, the DFSDM filter and integrator are automatically put into idle when not used by conversions (RCIP=0, JCIP=0).

32.5 DFSDM interrupts

In order to increase the CPU performance, a set of interrupts related to the CPU event occurrence has been implemented:

- End of injected conversion interrupt:
  - enabled by JEOCIE bit in DFSDM_FLTxCR2 register
  - indicated in JEOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxJDATAR register (injected data)
  - indication of which channel end of conversion occurred, reported in JDATACH[2:0] bits in DFSDM_FLTxJDATAR register
- End of regular conversion interrupt:
  - enabled by REOCIE bit in DFSDM_FLTxCR2 register
  - indicated in REOCF bit in DFSDM_FLTxISR register
  - cleared by reading DFSDM_FLTxRDATAR register (regular data)
  - indication of which channel end of conversion occurred, reported in RDATACH[2:0] bits in DFSDM_FLTxRDATAR register
- Data overrun interrupt for injected conversions:
  - occurred when injected converted data were not read from DFSDM_FLTxJDATAR register (by CPU or DMA) and were overwritten by a new injected conversion
  - enabled by JOVRIE bit in DFSDM_FLTxCR2 register
  - indicated in JOVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRJOVRF bit in DFSDM_FLTxICR register
- Data overrun interrupt for regular conversions:
  - occurred when regular converted data were not read from DFSDM_FLTxRDATAR register (by CPU or DMA) and were overwritten by a new regular conversion
  - enabled by ROVRIE bit in DFSDM_FLTxCR2 register
  - indicated in ROVRF bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRROVRF bit in DFSDM_FLTxICR register
- Analog watchdog interrupt:

- occurred when converted data (output data or data from analog watchdog filter - according to AWFSEL bit setting in DFSDM_FLTxCR1 register) crosses over/under high/low thresholds in DFSDM_FLTxAWHTR / DFSDM_FLTxAWLTR registers
- enabled by AWDIE bit in DFSDM_FLTxCR2 register (on selected channels AWDCH[7:0])
- indicated in AWDF bit in DFSDM_FLTxISR register
- separate indication of high or low analog watchdog threshold error by AWHTF[7:0] and AWLTF[7:0] fields in DFSDM_FLTxAWSR register
- cleared by writing ‘1’ into corresponding CLRAWHTF[7:0] or CLRAWLTF[7:0] bits in DFSDM_FLTxAWCFR register

- Short-circuit detector interrupt:
  - occurred when the number of stable data crosses over thresholds in DFSDM_CHyAWSCDR register
  - enabled by SCDIE bit in DFSDM_FLTxCR2 register (on channel selected by SCDEN bi tin DFSDM_CHyCFGR1 register)
  - indicated in SCDF[7:0] bits in DFSDM_FLTxISR register (which also reports the channel on which the short-circuit detector event occurred)
  - cleared by writing ‘1’ into the corresponding CLRSCDF[7:0] bit in DFSDM_FLTxICR register
- Channel clock absence interrupt:
  - occurred when there is clock absence on CKINy pin (see *Clock absence detection* in *Section 32.4.4: Serial channel transceivers*)
  - enabled by CKABIE bit in DFSDM_FLTxCR2 register (on channels selected by CKABEN bit in DFSDM_CHyCFGR1 register)
  - indicated in CKABF[y] bit in DFSDM_FLTxISR register
  - cleared by writing ‘1’ into CLRCKABF[y] bit in DFSDM_FLTxICR register

Table 227. DFSDM interrupt requests

Interrupt event	Event flag	Event/Interrupt clearing method	Interrupt enable control bit
End of injected conversion	JEOCF	reading DFSDM_FLTxJDATAR	JEOCIE
End of regular conversion	REOCF	reading DFSDM_FLTxRDATAR	REOCIE
Injected data overrun	JOVRF	writing CLRJOVRF = 1	JOVRIE
Regular data overrun	ROVRF	writing CLRROVRF = 1	ROVRIE
Analog watchdog	AWDF, AWHTF[7:0], AWLTF[7:0]	writing CLRAWHTF[7:0] = 1 writing CLRAWLTF[7:0] = 1	AWDIE, (AWDCH[7:0])
short-circuit detector	SCDF[7:0]	writing CLRSCDF[7:0] = 1	SCDIE, (SCDEN)
Channel clock absence	CKABF[7:0]	writing CLRCKABF[7:0] = 1	CKABIE, (CKABEN)

32.6 DFSDM DMA transfer

To decrease the CPU intervention, conversions can be transferred into memory using a DMA transfer. A DMA transfer for injected conversions is enabled by setting bit JDMAEN=1 in DFSDM_FLTxCR1 register. A DMA transfer for regular conversions is enabled by setting bit RDMAEN=1 in DFSDM_FLTxCR1 register.

Note: With a DMA transfer, the interrupt flag is automatically cleared at the end of the injected or regular conversion (JEOCF or REOCF bit in DFSDM_FLTxISR register) because DMA is reading DFSDM_FLTxJDATAR or DFSDM_FLTxRDATAR register.

32.7 DFSDM channel y registers (y=0..7)

Word access (32-bit) must be used for registers write access except DFSDM_CHyDATINR register. Write access to DFSDM_CHyDATINR register can be either word access (32-bit) or half-word access (16-bit).

32.7.1 DFSDM channel y configuration register (DFSDM_CHyCFGR1)

This register specifies the parameters used by channel y.

Address offset: 0x00 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFSDMEN	CKOUTSRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]							
rw	rw							rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
rw	rw	rw	rw				rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 **DFSDMEN**: Global enable for DFSDM interface
- 0: DFSDM interface disabled
- 1: DFSDM interface enabled

If DFSDM interface is enabled, then it is started to operate according to enabled y channels and enabled x filters settings (CHEN bit in DFSDM_CHyCFGR1 and DFEN bit in DFSDM_FLTxCR1).
Data cleared by setting DFSDMEN=0:
–all registers DFSDM_FLTxISR are set to reset state (x = 0..5)
–all registers DFSDM_FLTxAWSR are set to reset state (x = 0..5)

Note: DFSDMEN is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bit 30 **CKOUTSRC**: Output serial clock source selection
- 0: Source for output clock is from system clock
- 1: Source for output clock is from audio clock

This value can be modified only when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).

Note: CKOUTSRC is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **CKOUTDIV[7:0]**: Output serial clock divider

0: Output clock generation is disabled (CKOUT signal is set to low state)

1- 255: Defines the division of system clock for the serial clock output for CKOUT signal in range 2 - 256 (Divider = CKOUTDIV+1).

CKOUTDIV also defines the threshold for a clock absence detection.

This value can only be modified when DFSDMEN=0 (in DFSDM_CH0CFGR1 register).

If DFSDMEN=0 (in DFSDM_CH0CFGR1 register) then CKOUT signal is set to low state (setting is performed one DFSDM clock cycle after DFSDMEN=0).

Note: CKOUTDIV is present only in DFSDM_CH0CFGR1 register (channel y=0)

Bits 15:14 **DATPACK[1:0]**: Data packing mode in DFSDM_CHyDATINR register.

0: Standard: input data in DFSDM_CHyDATINR register are stored only in INDAT0[15:0]. To empty DFSDM_CHyDATINR register one sample must be read by the DFSDM filter from channel y.

1: Interleaved: input data in DFSDM_CHyDATINR register are stored as two samples:

–first sample in INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y)

To empty DFSDM_CHyDATINR register, two samples must be read by the digital filter from channel y (INDAT0[15:0] part is read as first sample and then INDAT1[15:0] part is read as next sample).

2: Dual: input data in DFSDM_CHyDATINR register are stored as two samples:

–first sample INDAT0[15:0] (assigned to channel y)

–second sample INDAT1[15:0] (assigned to channel y+1)

To empty DFSDM_CHyDATINR register first sample must be read by the digital filter from channel y and second sample must be read by another digital filter from channel y+1. Dual mode is available only on even channel numbers (y = 0, 2, 4, 6), for odd channel numbers (y = 1, 3, 5, 7) DFSDM_CHyDATINR is write protected. If an even channel is set to dual mode then the following odd channel must be set into standard mode (DATPACK[1:0]=0) for correct cooperation with even channel.

3: Reserved

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 13:12 **DATMPX[1:0]**: Input data multiplexer for channel y

0: Data to channel y are taken from external serial inputs as 1-bit values. DFSDM_CHyDATINR register is write protected.

1: Data to channel y are taken from internal analog to digital converter ADC_{y+1} output register update as 16-bit values (if ADC_{y+1} is available). Data from ADCs are written into INDAT0[15:0] part of DFSDM_CHyDATINR register.

2: Data to channel y are taken from internal DFSDM_CHyDATINR register by direct CPU/DMA write. There can be written one or two 16-bit data samples according DATPACK[1:0] bit field setting.

3: Reserved

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **CHINSEL**: Channel inputs selection

0: Channel inputs are taken from pins of the same channel y.

1: Channel inputs are taken from pins of the following channel (channel (y+1) modulo 8).

This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 7 **CHEN**: Channel y enable

0: Channel y disabled

1: Channel y enabled

If channel y is enabled, then serial data receiving is started according to the given channel setting.

Bit 6 **CKABEN**: Clock absence detector enable on channel y
0: Clock absence detector disabled on channel y
1: Clock absence detector enabled on channel y

Bit 5 **SCDEN**: Short-circuit detector enable on channel y
0: Input channel y will not be guarded by the short-circuit detector
1: Input channel y will be continuously guarded by the short-circuit detector

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **SPICKSEL[1:0]**: SPI clock select for channel y
0: clock coming from external CKINy input - sampling point according SITP[1:0]
1: clock coming from internal CKOUT output - sampling point according SITP[1:0]
2: clock coming from internal CKOUT - sampling point on each second CKOUT falling edge.
For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input rising edge).
3: clock coming from internal CKOUT output - sampling point on each second CKOUT rising edge.
For connection to external ΣΔ modulator which divides its clock input (from CKOUT) by 2 to generate its output serial communication clock (and this output clock change is active on each clock input falling edge).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 1:0 **SITP[1:0]**: Serial interface type for channel y
00: SPI with rising edge to strobe data
01: SPI with falling edge to strobe data
10: Manchester coded input on DATINy pin: rising edge = logic 0, falling edge = logic 1
11: Manchester coded input on DATINy pin: rising edge = logic 1, falling edge = logic 0
This value can only be modified when CHEN=0 (in DFSDM_CHyCFGR1 register).

32.7.2 DFSDM channel y configuration register (DFSDM_CHyCFGR2)

This register specifies the parameters used by channel y.

Address offset: 0x04 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OFFSET[7:0]								DTRBS[4:0]					Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:8 **OFFSET[23:0]**: 24-bit calibration offset for channel y
For channel y, OFFSET is applied to the results of each conversion from this channel.
This value is set by software.

Bits 7:3 **DTRBS[4:0]**: Data right bit-shift for channel y
0-31: Defines the shift of the data result coming from the integrator - how many bit shifts to the right will be performed to have final results. Bit-shift is performed before offset correction. The data shift is rounding the result to nearest integer value. The sign of shifted result is maintained (to have valid 24-bit signed format of result data).
This value can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bits 2:0 Reserved, must be kept at reset value.

32.7.3 DFSDM channel y analog watchdog and short-circuit detector register (DFSDM_CHyAWSCDR)

Short-circuit detector and analog watchdog settings for channel y.

Address offset: 0x08 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]				
								rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **AWFORD[1:0]**: Analog watchdog Sinc filter order on channel y
0: FastSinc filter type
1: $Sinc^1$ filter type
2: $Sinc^2$ filter type
3: $Sinc^3$ filter type
$Sinc^x$ filter type transfer function:

$$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$

FastSinc filter type transfer function:

$$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot FOSR)})$$

This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **AWFOSR[4:0]**: Analog watchdog filter oversampling ratio (decimation rate) on channel y
0 - 31: Defines the length of the Sinc type filter in the range 1 - 32 (AWFOSR + 1). This number is also the decimation ratio of the analog data rate.
This bit can be modified only when CHEN=0 (in DFSDM_CHyCFGR1 register).

Note: If AWFOSR = 0 then the filter has no effect (filter bypass).

Bits 15:12 **BKSCD[3:0]**: Break signal assignment for short-circuit detector on channel y
BKSCD[i] = 0: Break i signal not assigned to short-circuit detector on channel y
BKSCD[i] = 1: Break i signal assigned to short-circuit detector on channel y

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:0 **SCDT[7:0]**: short-circuit detector threshold for channel y
These bits are written by software to define the threshold counter for the short-circuit detector. If this value is reached, then a short-circuit detector event occurs on a given channel.

32.7.4 DFSDM channel y watchdog filter data register (DFSDM_CHyWDATR)

This register contains the data resulting from the analog watchdog filter associated to the input channel y.

Address offset: 0x0C + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WDATA[15:0]**: Input channel y watchdog data
Data converted by the analog watchdog filter for input channel y. This data is continuously converted (no trigger) for this channel, with a limited resolution (OSR=1..32/sinc order = 1..3).

32.7.5 DFSDM channel y data input register (DFSDM_CHyDATINR)

This register contains 16-bit input data to be processed by DFSDM filter module. Write access can be either word access (32-bit) or half-word access (16-bit).

Address offset: 0x10 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INDAT1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INDAT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INDAT1[15:0]**: Input data for channel y or channel y+1

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2. Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).

If DATPACK[1:0]=0 (standard mode)

INDAT0[15:0] is write protected (not used for input sample).

If DATPACK[1:0]=1 (interleaved mode)

Second channel y data sample is stored into INDAT1[15:0]. First channel y data sample is stored into INDAT0[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.

If DATPACK[1:0]=2 (dual mode).

For even y channels: sample in INDAT1[15:0] is automatically copied into INDAT0[15:0] of channel (y+1).

For odd y channels: INDAT1[15:0] is write protected.

See *Section 32.4.6: Parallel data inputs* for more details.

INDAT0[15:1] is in the16-bit signed format.

Bits 15:0 **INDAT0[15:0]**: Input data for channel y

Input parallel channel data to be processed by the digital filter if DATMPX[1:0]=1 or DATMPX[1:0]=2. Data can be written by CPU/DMA (if DATMPX[1:0]=2) or directly by internal ADC (if DATMPX[1:0]=1).

If DATPACK[1:0]=0 (standard mode)

Channel y data sample is stored into INDAT0[15:0].

If DATPACK[1:0]=1 (interleaved mode)

First channel y data sample is stored into INDAT0[15:0]. Second channel y data sample is stored into INDAT1[15:0]. Both samples are read sequentially by DFSDM_FLTx filter as two channel y data samples.

If DATPACK[1:0]=2 (dual mode).

For even y channels: Channel y data sample is stored into INDAT0[15:0].

For odd y channels: INDAT0[15:0] is write protected.

See *Section 32.4.6: Parallel data inputs* for more details.

INDAT0[15:0] is in the16-bit signed format.

32.7.6 DFSDM channel y delay register (DFSDM_CHyDLYR)

Address offset: 0x14 + 0x20 * y, (y = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **PLSSKP[5:0]**: Pulses to skip for input data skipping function

0-63: Defines the number of serial input samples that will be skipped. Skipping is applied immediately after writing to this field. Reading of PLSSKP[5:0] returns current value of pulses which will be skipped. If PLSSKP[5:0]=0 then all required data samples were already skipped.

Note: User can update PLSSKP[5:0] also when PLSSKP[5:0] is not zero.

32.8 DFSDM filter x module registers (x=0..5)

Word access (32-bit) must be used for registers write access except DFSDM_CHyDATINR register.

32.8.1 DFSDM filter x control register 1 (DFSDM_FLTxCR1)

Address offset: 0x100 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSWSTART	Res.
	rw	rw			rw	rw	rw			rw		rw	rw	rt_w1	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSWSTART	DFEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw		rt_w1	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **AWFSEL**: Analog watchdog fast mode select

0: Analog watchdog on data output value (after the digital filter). The comparison is done after offset correction and shift

1: Analog watchdog on channel transceivers value (after watchdog filter)

Bit 29 **FAST**: Fast conversion mode selection for regular conversions

0: Fast conversion mode disabled

1: Fast conversion mode enabled

When converting a regular conversion in continuous mode, having enabled the fast mode causes each conversion (except the first) to execute faster than in standard mode. This bit has no effect on conversions which are not continuous.

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

if FAST=0 (or first conversion in continuous mode if FAST=1):

$t = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$..... for Sinc^x filters

$t = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$..... for FastSinc filter

if FAST=1 in continuous mode (except first conversion):

$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if $F_{OSR} = F_{OSR}[9:0]+1 = 1$ (filter bypassed, active only integrator):

$t = I_{OSR} / f_{CKIN}$ (... but CNVCNT=0)

where: f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input.

Bits 28:27 Reserved, must be kept at reset value.

Bits 26:24 **RCH[2:0]**: Regular channel selection
0: Channel 0 is selected as the regular channel
1: Channel 1 is selected as the regular channel
...
7: Channel 7 is selected as the regular channel
Writing these bits when RCIP=1 takes effect when the next regular conversion begins. This is especially useful in continuous mode (when RCONT=1). It also affects regular conversions which are pending (due to ongoing injected conversion).

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RDMAEN**: DMA channel enabled to read data for the regular conversion
0: The DMA channel is not enabled to read regular data
1: The DMA channel is enabled to read regular data
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 20 Reserved, must be kept at reset value.

Bit 19 **RSYNC**: Launch regular conversion synchronously with DFSDM_FLT0
0: Do not launch a regular conversion synchronously with DFSDM_FLT0
1: Launch a regular conversion in this DFSDM_FLTx at the very moment when a regular conversion is launched in DFSDM_FLT0
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 18 **RCONT**: Continuous mode selection for regular conversions
0: The regular channel is converted just once for each conversion request
1: The regular channel is converted repeatedly after each conversion request
Writing '0' to this bit while a continuous regular conversion is already in progress stops the continuous mode immediately.

Bit 17 **RSWSTART**: Software start of a conversion on the regular channel
0: Writing '0' has no effect
1: Writing '1' makes a request to start a conversion on the regular channel and causes RCIP to become '1'. If RCIP=1 already, writing to RSWSTART has no effect. Writing '1' has no effect if RSYNC=1.
This bit is always read as '0'.

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:13 **JEXTEN[1:0]**: Trigger enable and trigger edge selection for injected conversions
00: Trigger detection is disabled
01: Each rising edge on the selected trigger makes a request to launch an injected conversion
10: Each falling edge on the selected trigger makes a request to launch an injected conversion
11: Both rising edges and falling edges on the selected trigger make requests to launch injected conversions
This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bits 12:8 **JEXTSEL[4:0]**: Trigger signal selection for launching injected conversions

0x0-0x1F: Trigger inputs selected by the following table (internal or external trigger).

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Note: synchronous trigger has latency up to one $f_{DFSDMCLK}$ clock cycle (with deterministic jitter), asynchronous trigger has latency 2-3 $f_{DFSDMCLK}$ clock cycles (with jitter up to 1 cycle).

	DFSDM_FLTx
0x00	dfsdm_jtrg0
0x01	dfsdm_jtrg1
...	
0x1E	dfsdm_jtrg30
0x1F	dfsdm_jtrg31

Refer to *Table 223: DFSDM triggers connection*.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **JDMAEN**: DMA channel enabled to read data for the injected channel group

0: The DMA channel is not enabled to read injected data

1: The DMA channel is enabled to read injected data

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 4 **JSCAN**: Scanning conversion mode for injected conversions

0: One channel conversion is performed from the injected channel group and next the selected channel from this group is selected.

1: The series of conversions for the injected group channels is executed, starting over with the lowest selected channel.

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Writing JCHG if JSCAN=0 resets the channel selection to the lowest selected channel.

Bit 3 **JSYNC**: Launch an injected conversion synchronously with the DFSDM_FLT0 JSWSTART trigger

0: Do not launch an injected conversion synchronously with DFSDM_FLT0

1: Launch an injected conversion in this DFSDM_FLTx at the very moment when an injected conversion is launched in DFSDM_FLT0 by its JSWSTART trigger

This bit can be modified only when DFEN=0 (DFSDM_FLTxCR1).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **JSWSTART**: Start a conversion of the injected group of channels

0: Writing ‘0’ has no effect.

1: Writing ‘1’ makes a request to convert the channels in the injected conversion group, causing JCIP to become ‘1’ at the same time. If JCIP=1 already, then writing to JSWSTART has no effect. Writing ‘1’ has no effect if JSYNC=1.

This bit is always read as ‘0’.

Bit 0 **DFEN**: DFSDM_FLTx enable

0: DFSDM_FLTx is disabled. All conversions of given DFSDM_FLTx are stopped immediately and all DFSDM_FLTx functions are stopped.

1: DFSDM_FLTx is enabled. If DFSDM_FLTx is enabled, then DFSDM_FLTx starts operating according to its setting.

Data which are cleared by setting DFEN=0:

–register DFSDM_FLTxISR is set to the reset state

–register DFSDM_FLTxAWSR is set to the reset state

32.8.2 DFSDM filter x control register 2 (DFSDM_FLTxCR2)

Address offset: 0x104 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXCH[7:0]								Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **AWDCH[7:0]**: Analog watchdog channel selection

These bits select the input channel to be guarded continuously by the analog watchdog.
AWDCH[y] = 0: Analog watchdog is disabled on channel y
AWDCH[y] = 1: Analog watchdog is enabled on channel y

Bits 15:8 **EXCH[7:0]**: Extremes detector channel selection

These bits select the input channels to be taken by the Extremes detector.
EXCH[y] = 0: Extremes detector does not accept data from channel y
EXCH[y] = 1: Extremes detector accepts data from channel y

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CKABIE**: Clock absence interrupt enable

0: Detection of channel input clock absence interrupt is disabled
1: Detection of channel input clock absence interrupt is enabled
Please see the explanation of CKABF[7:0] in DFSDM_FLTxISR.

Note: CKABIE is present only in DFSDM_FLT0CR2 register (filter x=0)

Bit 5 **SCDIE**: Short-circuit detector interrupt enable

0: short-circuit detector interrupt is disabled
1: short-circuit detector interrupt is enabled
Please see the explanation of SCDF[7:0] in DFSDM_FLTxISR.

Note: SCDIE is present only in DFSDM_FLT0CR2 register (filter x=0)

Bit 4 **AWDIE**: Analog watchdog interrupt enable

0: Analog watchdog interrupt is disabled
1: Analog watchdog interrupt is enabled
Please see the explanation of AWDF in DFSDM_FLTxISR.

Bit 3 **ROVRIE**: Regular data overrun interrupt enable

0: Regular data overrun interrupt is disabled
1: Regular data overrun interrupt is enabled
Please see the explanation of ROVRF in DFSDM_FLTxISR.

Bit 2 **JOVRIE**: Injected data overrun interrupt enable
0: Injected data overrun interrupt is disabled
1: Injected data overrun interrupt is enabled
Please see the explanation of JOVRF in DFSDM_FLTxISR.

Bit 1 **REOCIE**: Regular end of conversion interrupt enable
0: Regular end of conversion interrupt is disabled
1: Regular end of conversion interrupt is enabled
Please see the explanation of REOCF in DFSDM_FLTxISR.

Bit 0 **JEOCIE**: Injected end of conversion interrupt enable
0: Injected end of conversion interrupt is disabled
1: Injected end of conversion interrupt is enabled
Please see the explanation of JEOCF in DFSDM_FLTxISR.

32.8.3 DFSDM filter x interrupt and status register (DFSDM_FLTxISR)

Address offset: 0x108 + 0x80 * x, (x = 0 to 5)

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCDF[7:0]								CKABF[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	r	r									r	r	r	r	r

Bits 31:24 **SCDF[7:0]**: short-circuit detector flag
SDCF[y]=0: No short-circuit detector event occurred on channel y
SDCF[y]=1: The short-circuit detector counter reaches, on channel y, the value programmed in the DFSDM_CHyAWSCDR registers
This bit is set by hardware. It can be cleared by software using the corresponding CLRSCDF[y] bit in the DFSDM_FLTxICR register. SCDF[y] is cleared also by hardware when CHEN[y] = 0 (given channel is disabled).
Note: SCDF[7:0] is present only in DFSDM_FLT0ISR register (filter x=0)

Bits 23:16 **CKABF[7:0]**: Clock absence flag
CKABF[y]=0: Clock signal on channel y is present.
CKABF[y]=1: Clock signal on channel y is not present.
Given y bit is set by hardware when clock absence is detected on channel y. It is held at CKABF[y]=1 state by hardware when CHEN=0 (see DFSDM_CHyCFGR1 register). It is held at CKABF[y]=1 state by hardware when the transceiver is not yet synchronized.It can be cleared by software using the corresponding CLRCKABF[y] bit in the DFSDM_FLTxICR register.
Note: CKABF[7:0] is present only in DFSDM_FLT0ISR register (filter x=0)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RCIP**: Regular conversion in progress status
0: No request to convert the regular channel has been issued
1: The conversion of the regular channel is in progress or a request for a regular conversion is pending
A request to start a regular conversion is ignored when RCIP=1.

Bit 13 **JCIP**: Injected conversion in progress status

0: No request to convert the injected channel group (neither by software nor by trigger) has been issued

1: The conversion of the injected channel group is in progress or a request for a injected conversion is pending, due either to ‘1’ being written to JSWSTART or to a trigger detection

A request to start an injected conversion is ignored when JCIP=1.

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **AWDF**: Analog watchdog

0: No Analog watchdog event occurred

1: The analog watchdog block detected voltage which crosses the value programmed in the DFSDM_FLTxAWLTR or DFSDM_FLTxAWHTR registers.

This bit is set by hardware. It is cleared by software by clearing all source flag bits AWHTF[7:0] and AWLTF[7:0] in DFSDM_FLTxAWSR register (by writing ‘1’ into the clear bits in DFSDM_FLTxAWCFR register).

Bit 3 **ROVRF**: Regular conversion overrun flag

0: No regular conversion overrun has occurred

1: A regular conversion overrun has occurred, which means that a regular conversion finished while REOCF was already ‘1’. RDATAR is not affected by overruns

This bit is set by hardware. It can be cleared by software using the CLRROVRF bit in the DFSDM_FLTxICR register.

Bit 2 **JOVRF**: Injected conversion overrun flag

0: No injected conversion overrun has occurred

1: An injected conversion overrun has occurred, which means that an injected conversion finished while JEOCF was already ‘1’. JDATAR is not affected by overruns

This bit is set by hardware. It can be cleared by software using the CLRJOVRF bit in the DFSDM_FLTxICR register.

Bit 1 **REOCF**: End of regular conversion flag

0: No regular conversion has completed

1: A regular conversion has completed and its data may be read

This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxRDATAR.

Bit 0 **JEOCF**: End of injected conversion flag

0: No injected conversion has completed

1: An injected conversion has completed and its data may be read

This bit is set by hardware. It is cleared when the software or DMA reads DFSDM_FLTxJDATAR.

Note: For each of the flag bits, an interrupt can be enabled by setting the corresponding bit in DFSDM_FLTxCR2. If an interrupt is called, the flag must be cleared before exiting the interrupt service routine.

All the bits of DFSDM_FLTxISR are automatically reset when DFEN=0.

32.8.4 DFSDM filter x interrupt flag clear register (DFSDM_FLTxICR)

Address offset: 0x10C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRSCDF[7:0]								CLRCKABF[7:0]							
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRR OVRF	CLRJ OVRF	Res.	Res.
												rc_w1	rc_w1		

Bits 31:24 **CLRSCDF[7:0]**: Clear the short-circuit detector flag

CLRSCDF[y]=0: Writing ‘0’ has no effect

CLRSCDF[y]=1: Writing ‘1’ to position y clears the corresponding SCDF[y] bit in the DFSDM_FLTxISR register

Note: CLRSCDF[7:0] is present only in DFSDM_FLT0ICR register (filter x=0)

Bits 23:16 **CLRCKABF[7:0]**: Clear the clock absence flag

CLRCKABF[y]=0: Writing ‘0’ has no effect

CLRCKABF[y]=1: Writing ‘1’ to position y clears the corresponding CKABF[y] bit in the DFSDM_FLTxISR register. When the transceiver is not yet synchronized, the clock absence flag is set and cannot be cleared by CLRCKABF[y].

Note: CLRCKABF[7:0] is present only in DFSDM_FLT0ICR register (filter x=0)

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CLRROVRF**: Clear the regular conversion overrun flag

0: Writing ‘0’ has no effect

1: Writing ‘1’ clears the ROVRF bit in the DFSDM_FLTxISR register

Bit 2 **CLRJOVRF**: Clear the injected conversion overrun flag

0: Writing ‘0’ has no effect

1: Writing ‘1’ clears the JOVRF bit in the DFSDM_FLTxISR register

Bits 1:0 Reserved, must be kept at reset value.

Note: The bits of DFSDM_FLTxICR are always read as ‘0’.

32.8.5 DFSDM filter x injected channel group selection register (DFSDM_FLTxJCHGR)

Address offset: 0x110 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **JCHG[7:0]**: Injected channel group selection

JCHG[y]=0: channel y is not part of the injected group

JCHG[y]=1: channel y is part of the injected group

If JSCAN=1, each of the selected channels is converted, one after another. The lowest channel (channel 0, if selected) is converted first and the sequence ends at the highest selected channel.

If JSCAN=0, then only one channel is converted from the selected channels, and the channel selection is moved to the next channel. Writing JCHG, if JSCAN=0, resets the channel selection to the lowest selected channel.

At least one channel must always be selected for the injected group. Writes causing all JCHG bits to be zero are ignored.

32.8.6 DFSDM filter x control register (DFSDM_FLTxFCR)

Address offset: 0x114 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]									
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **FORD[2:0]**: Sinc filter order

0: FastSinc filter type
1: $Sinc^1$ filter type
2: $Sinc^2$ filter type
3: $Sinc^3$ filter type
4: $Sinc^4$ filter type
5: $Sinc^5$ filter type
6-7: Reserved
$Sinc^x$ filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^x$$

FastSinc filter type transfer function: $$H(z) = \left(\frac{1 - z^{-FOSR}}{1 - z^{-1}}\right)^2 \cdot \left(1 + z^{-(2 \cdot FOSR)}\right)$$

This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1).

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:16 **FOSR[9:0]**: Sinc filter oversampling ratio (decimation rate)

0 - 1023: Defines the length of the Sinc type filter in the range 1 - 1024 (F_{OSR} = FOSR[9:0] +1). This number is also the decimation ratio of the output data rate from filter.
This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1)

Note: If FOSR = 0, then the filter has no effect (filter bypass).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IOSR[7:0]**: Integrator oversampling ratio (averaging length)

0- 255: The length of the Integrator in the range 1 - 256 (IOSR + 1). Defines how many samples from Sinc filter will be summed into one output data sample from the integrator. The output data rate from the integrator will be decreased by this number (additional data decimation ratio).
This bit can only be modified when DFEN=0 (DFSDM_FLTxCR1)

Note: If IOSR = 0, then the Integrator has no effect (Integrator bypass).

32.8.7 DFSDM filter x data register for injected group (DFSDM_FLTxJDATAR)

Address offset: 0x118 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
JDATA[7:0]								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:8 **JDATA[23:0]**: Injected group conversion data

When each conversion of a channel in the injected group finishes, its resulting data is stored in this field. The data is valid when JEOCF=1. Reading this register clears the corresponding JEOCF.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **JDATACH[2:0]**: Injected channel most recently converted

When each conversion of a channel in the injected group finishes, JDATACH[2:0] is updated to indicate which channel was converted. Thus, JDATA[23:0] holds the data that corresponds to the channel indicated by JDATACH[2:0].

Note: DMA may be used to read the data from this register. Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears JEOCF in DFSDM_FLTxISR. Thus, the firmware must not read this register if DMA is activated to read data from this register.

32.8.8 DFSDM filter x data register for the regular channel (DFSDM_FLTxRDATAR)

Address offset: 0x11C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDATA[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[7:0]								Res.	Res.	Res.	RPEND	Res.	RDATACH[2:0]		
r	r	r	r	r	r	r	r				r		r	r	r

Bits 31:8 **RDATA[23:0]**: Regular channel conversion data

When each regular conversion finishes, its data is stored in this register. The data is valid when REOCF=1. Reading this register clears the corresponding REOCF.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **RPEND**: Regular channel pending data

Regular data in RDATA[23:0] was delayed due to an injected channel trigger during the conversion

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **RDATACH[2:0]**: Regular channel most recently converted

When each regular conversion finishes, RDATACH[2:0] is updated to indicate which channel was converted (because regular channel selection RCH[2:0] in DFSDM_FLTxCR1 register can be updated during regular conversion). Thus RDATA[23:0] holds the data that corresponds to the channel indicated by RDATACH[2:0].

Note: Half-word accesses may be used to read only the MSBs of conversion data.

Reading this register also clears REOCF in DFSDM_FLTxISR.

32.8.9 DFSDM filter x analog watchdog high threshold register (DFSDM_FLTxAWHTR)

Address offset: 0x120 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWHT[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AWHT[7:0]								Res.	Res.	Res.	Res.	BKAWH[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:8 **AWHT[23:0]**: Analog watchdog high threshold

These bits are written by software to define the high threshold for the analog watchdog.

Note: In case channel transceivers monitor (AWFSEL=1), the higher 16 bits (AWHT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWHT[7:0] are not taken into comparison in this case.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **BKAWH[3:0]**: Break signal assignment to analog watchdog high threshold event

BKAWH[i] = 0: Break i signal is not assigned to an analog watchdog high threshold event

BKAWH[i] = 1: Break i signal is assigned to an analog watchdog high threshold event

32.8.10 DFSDM filter x analog watchdog low threshold register (DFSDM_FLTxAWLTR)

Address offset: 0x124 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AWLT[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AWLT[7:0]								Res.	Res.	Res.	Res.	BKAWL[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:8 **AWLT[23:0]**: Analog watchdog low threshold

These bits are written by software to define the low threshold for the analog watchdog.

Note: In case channel transceivers monitor (AWFSEL=1), only the higher 16 bits (AWLT[23:8]) define the 16-bit threshold as compared with the analog watchdog filter output (because data coming from the analog watchdog filter are up to a 16-bit resolution). Bits AWLT[7:0] are not taken into comparison in this case.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **BKAWL[3:0]**: Break signal assignment to analog watchdog low threshold event

BKAWL[i] = 0: Break i signal is not assigned to an analog watchdog low threshold event

BKAWL[i] = 1: Break i signal is assigned to an analog watchdog low threshold event

32.8.11 DFSDM filter x analog watchdog status register (DFSDM_FLTxAWSR)

Address offset: 0x128 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AWHTF[7:0]								AWLTF[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **AWHTF[7:0]**: Analog watchdog high threshold flag

AWHTF[y]=1 indicates a high threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWHTF[y] bit in the DFSDM_FLTxAWCFR register.

Bits 7:0 **AWLTF[7:0]**: Analog watchdog low threshold flag

AWLTF[y]=1 indicates a low threshold error on channel y. It is set by hardware. It can be cleared by software using the corresponding CLRAWLTF[y] bit in the DFSDM_FLTxAWCFR register.

Note: All the bits of DFSDM_FLTxAWSR are automatically reset when DFEN=0.

32.8.12 DFSDM filter x analog watchdog clear flag register (DFSDM_FLTxAWCFR)

Address offset: 0x12C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLRAWHTF[7:0]								CLRAWLTF[7:0]							
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **CLRAWHTF[7:0]**: Clear the analog watchdog high threshold flag
CLRAWHTF[y]=0: Writing ‘0’ has no effect
CLRAWHTF[y]=1: Writing ‘1’ to position y clears the corresponding AWHTF[y] bit in the DFSDM_FLTxAWSR register

Bits 7:0 **CLRAWLTF[7:0]**: Clear the analog watchdog low threshold flag
CLRAWLTF[y]=0: Writing ‘0’ has no effect
CLRAWLTF[y]=1: Writing ‘1’ to position y clears the corresponding AWLTF[y] bit in the DFSDM_FLTxAWSR register

32.8.13 DFSDM filter x extremes detector maximum register (DFSDM_FLTxEXMAX)

Address offset: 0x130 + 0x80 * x, (x = 0 to 5)

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXMAX[23:8]															
rs_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXMAX[7:0]								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r						r	r	r

Bits 31:8 **EXMAX[23:0]**: Extremes detector maximum value
These bits are set by hardware and indicate the highest value converted by DFSDM_FLTx.
EXMAX[23:0] bits are reset to value (0x800000) by reading of this register.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EXMAXCH[2:0]**: Extremes detector maximum data channel.
These bits contains information about the channel on which the data is stored into EXMAX[23:0].
Bits are cleared by reading of this register.

32.8.14 DFSDM filter x extremes detector minimum register (DFSDM_FLTxEXMIN)

Address offset: 0x134 + 0x80 * x, (x = 0 to 5)

Reset value: 0x7FFF FF00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXMIN[23:8]															
rc_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXMIN[7:0]								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r	rs_r						r	r	r

Bits 31:8 **EXMIN[23:0]**: Extremes detector minimum value

These bits are set by hardware and indicate the lowest value converted by DFSDM_FLTx. EXMIN[23:0] bits are reset to value (0x7FFFFF) by reading of this register.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EXMINCH[2:0]**: Extremes detector minimum data channel

These bits contain information about the channel on which the data is stored into EXMIN[23:0]. Bits are cleared by reading of this register.

32.8.15 DFSDM filter x conversion timer register (DFSDM_FLTxCNVTIMR)

Address offset: 0x138 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNVCNT[27:12]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNVCNT[11:0]												Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r				

Bits 31:4 **CNVCNT[27:0]**: 28-bit timer counting conversion time t = CNVCNT[27:0] / $f_{DFSDMCLK}$

The timer has an input clock from DFSDM clock (system clock $f_{DFSDMCLK}$). Conversion time measurement is started on each conversion start and stopped when conversion finishes (interval between first and last serial sample). Only in case of filter bypass (FOSR[9:0] = 0) is the conversion time measurement stopped and CNVCNT[27:0] = 0. The counted time is:

if FAST=0 (or first conversion in continuous mode if FAST=1):

$t = [F_{OSR} * (I_{OSR}-1 + F_{ORD}) + F_{ORD}] / f_{CKIN}$..... for $Sinc^x$ filters

$t = [F_{OSR} * (I_{OSR}-1 + 4) + 2] / f_{CKIN}$..... for FastSinc filter

if FAST=1 in continuous mode (except first conversion):

$t = [F_{OSR} * I_{OSR}] / f_{CKIN}$

in case if F_{OSR} = FOSR[9:0]+1 = 1 (filter bypassed, active only integrator):

CNVCNT = 0 (counting is stopped, conversion time: $t = I_{OSR} / f_{CKIN}$)

where: f_{CKIN} is the channel input clock frequency (on given channel CKINy pin) or input data rate in case of parallel data input (from internal ADC or from CPU/DMA write)

Note: When conversion is interrupted (e.g. by disable/enable selected channel) the timer counts also this interruption time.

Bits 3:0 Reserved, must be kept at reset value.

32.9 DFSDM version registers

32.9.1 DFSDM hardware configuration register (DFSDM_HWCFGR)

This register specifies the hardware configuration of DFSDM peripheral.

Address offset: 0x7F0

Reset value: 0x0000 0608

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NBF[7:0]								NBT[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **NBF[7:0]**: Number of implemented filters
Defines how many filters are implemented in DFSDM peripheral.

Bits 7:0 **NBT[7:0]**: Number of implemented transceivers
Defines how many transceivers (input channels) are implemented in DFSDM peripheral.

32.9.2 DFSDM version register (DFSDM_VERR)

This register specifies the version of DFSDM peripheral.

Address offset: 0x7F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision of the DFSDM peripheral
These bits return the DFSDM major revision (in range 0..15).

Bits 3:0 **MINREV[3:0]**: Minor revision of the DFSDM peripheral
These bits return the DFSDM minor revision (in range 0..15).

32.9.3 DFSDM identification register (DFSDM_IPIDR)

This register specifies the identification of DFSDM peripheral.

Address offset: 0x7F8

Reset value: 0x0011 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Peripheral identifier
Bits [31:0]: these bits returns the DFSDM identifier ID[31:0] = 0x0011 0031

32.9.4 DFSDM size identification register (DFSDM_SIDR)

This register specifies the size allocated to DFSDM registers.

Address offset: 0x7FC

Reset value: 0xA3C5 DD02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification of DFSDM peripheral
Bits [31:8]: fixed code = 0xA3C5DD
Bits [7:0]: these bits returns the size of the memory region allocated to DFSDM registers.
0x02: 2KB allocated by DFSDM peripheral (fixed value).

32.9.5 DFSDM register map

The following table summarizes the DFSDM registers.

Table 228. DFSDM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DFSDM_ CH0CFGR1	DFSDMEN	CKOUTSRC	Res.	Res.	Res.	Res.	Res.	Res.	CKOUTDIV[7:0]								DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL [1:0]		SITP[1:0]	
	reset value	0	0							0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0		0	0	0	0
0x04	DFSDM_ CH0CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0																								0							
0x08	DFSDM_ CH0AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD [1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x0C	DFSDM_ CH0WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	DFSDM_ CH0DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	DFSDM_ CH0DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x18 - 0x1C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x20	DFSDM_ CH1CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL [1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x24	DFSDM_ CH1CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x28	DFSDM_ CH1AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x2C	DFSDM_ CH1WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	DFSDM_ CH1DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	DFSDM_ CH1DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38 - 0x3C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x40	DFSDM_ CH2CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x44	DFSDM_ CH2CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x48	DFSDM_ CH2AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x4C	DFSDM_ CH2WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	DFSDM_ CH2DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	DFSDM_ CH2DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x58 - 0x5C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x60	DFSDM_ CH3CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x64	DFSDM_ CH3CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x68	DFSDM_ CH3AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x6C	DFSDM_ CH3WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	DFSDM_ CH3DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	DFSDM_ CH3DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x78 - 0x7C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x80	DFSDM_ CH4CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0x84	DFSDM_ CH4CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x88	DFSDM_ CH4AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0x8C	DFSDM_ CH4WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	DFSDM_ CH4DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	DFSDM_ CH4DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0x98 - 0x9C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xA0	DFSDM_ CH5CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0xA4	DFSDM_ CH5CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xA8	DFSDM_ CH5AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0xAC	DFSDM_ CH5WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB0	DFSDM_ CH5DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	DFSDM_ CH5DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB8 - 0xBC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xC0	DFSDM_ CH6CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0xC4	DFSDM_ CH6CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xC8	DFSDM_ CH6AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0xCC	DFSDM_ CH6WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	DFSDM_ CH6DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD4	DFSDM_ CH6DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0
0xD8 - 0xDC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xE0	DFSDM_ CH7CFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATPACK[1:0]		DATMPX[1:0]		Res.	Res.	Res.	CHINSEL	CHEN	CKABEN	SCDEN	Res.	SPICKSEL[1:0]		SITP[1:0]	
	reset value																	0	0	0	0				0	0	0	0		0	0	0	0
0xE4	DFSDM_ CH7CFGR2	OFFSET[23:0]																								DTRBS[4:0]					Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0xE8	DFSDM_ CH7AWSCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWFORD[1:0]		Res.	AWFOSR[4:0]					BKSCD[3:0]				Res.	Res.	Res.	Res.	SCDT[7:0]							
	reset value									0	0		0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0
0xEC	DFSDM_ CH7WDATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF0	DFSDM_ CH7DATINR	INDAT1[15:0]																INDAT0[15:0]															
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF4	DFSDM_ CH7DLYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLSSKP[5:0]					
	reset value																											0	0	0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF8 - 0xFC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	DFSDM_ FLT0CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x104	DFSDM_ FLT0CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	CKABIE	SCDIE	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x108	DFSDM_ FLT0ISR	SCDF[7:0]								CKABF[7:0]								Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		0	0									0	0	0	0	0
0x10C	DFSDM_ FLT0ICR	CLRSCDF[7:0]								CLRCKABF[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0													0	0		
0x110	DFSDM_ FLT0JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x114	DFSDM_ FLT0FCR	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x118	DFSDM_ FLT0JDATAR	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH [2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x11C	DFSDM_ FLT0RDATAR	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x120	DFSDM_ FLT0AWHTR	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x124	DFSDM_ FLT0AWLTR	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x128	DFSDM_ FLT0AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C	DFSDM_ FLT0AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x130	DFSDM_ FLT0EXMAX	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x134	DFSDM_ FLT0EXMIN	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x138	DFSDM_ FLT0CNVTIMR	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x13C - 0x17C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x180	DFSDM_ FLT1CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x184	DFSDM_ FLT1CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x188	DFSDM_ FLT1ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x18C	DFSDM_ FLT1ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x190	DFSDM_ FLT1JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x194	DFSDM_ FLT1FCR	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x198	DFSDM_ FLT1JDATAR	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x19C	DFSDM_FLT1RDATAR	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x1A0	DFSDM_FLT1AWHTR	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x1A4	DFSDM_FLT1AWLTR	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x1A8	DFSDM_FLT1AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1AC	DFSDM_FLT1AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1B0	DFSDM_FLT1EXMAX	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x1B4	DFSDM_FLT1EXMIN	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x1B8	DFSDM_FLT1CNVTIMR	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x1BC - 0x1FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x200	DFSDM_FLT2CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x204	DFSDM_FLT2CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x208	DFSDM_FLT2ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x20C	DFSDM_FLT2ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x210	DFSDM_ FLT2JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x214	DFSDM_ FLT2FCR	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x218	DFSDM_ FLT2JDATAR	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x21C	DFSDM_ FLT2RDATAR	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x220	DFSDM_ FLT2AWHTR	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x224	DFSDM_ FLT2AWLTR	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x228	DFSDM_ FLT2AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x22C	DFSDM_ FLT2AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x230	DFSDM_ FLT2EXMAX	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x234	DFSDM_ FLT2EXMIN	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x238	DFSDM_ FLT2CNVTIMR	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x23C - 0x27C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x280	DFSDM_ FLT3CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x284	DFSDM_ FLT3CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x288	DFSDM_ FLT3ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x28C	DFSDM_ FLT3ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x290	DFSDM_ FLT3JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x294	DFSDM_ FLT3FCR	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x298	DFSDM_ FLT3JDATAR	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x29C	DFSDM_ FLT3RDATAR	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x2A0	DFSDM_ FLT3AWHTR	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x2A4	DFSDM_ FLT3AWLTR	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x2A8	DFSDM_ FLT3AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2AC	DFSDM_ FLT3AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2B0	DFSDM_ FLT3EXMAX	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2B4	DFSDM_ FLT3EXMIN	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x2B8	DFSDM_ FLT3CNVTIMR	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x2BC - 0x2FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	DFSDM_ FLT4CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x304	DFSDM_ FLT4CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x308	DFSDM_ FLT4ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x30C	DFSDM_ FLT4ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x310	DFSDM_ FLT4JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x314	DFSDM_ FLT4FCR	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x318	DFSDM_ FLT4JDATAR	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x31C	DFSDM_ FLT4RDATAR	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x320	DFSDM_ FLT4AWHTR	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x324	DFSDM_ FLT4AWLTR	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x328	DFSDM_ FLT4AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x32C	DFSDM_ FLT4AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x330	DFSDM_ FLT4EXMAX	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x334	DFSDM_ FLT4EXMIN	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x338	DFSDM_ FLT4CNVTIMR	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x33C - 0x37C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x380	DFSDM_ FLT5CR1	Res.	AWFSEL	FAST	Res.	Res.	RCH[2:0]			Res.	Res.	RDMAEN	Res.	RSYNC	RCONT	RSW START	Res.	Res.	JEXTEN[1:0]		JEXTSEL[4:0]					Res.	Res.	JDMAEN	JSCAN	JSYNC	Res.	JSW START	DFEN
	reset value		0	0			0	0	0			0		0	0	0			0	0	0	0	0	0	0			0	0	0		0	0
0x384	DFSDM_ FLT5CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[7:0]								EXCH[7:0]								Res.	Res.	Res.	AWDIE	ROVRIE	JOVRIE	REOCIE	JEOCIE
	reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0
0x388	DFSDM_ FLT5ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCIP	JCIP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDF	ROVRF	JOVRF	REOCF	JEOCF
	reset value																		0	0									0	0	0	0	0
0x38C	DFSDM_ FLT5ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR ROVRF	CLR JOVRF	Res.	Res.
	reset value																													0	0		
0x390	DFSDM_ FLT5JCHGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JCHG[7:0]							
	reset value																									0	0	0	0	0	0	0	1
0x394	DFSDM_ FLT5FCR	FORD[2:0]			Res.	Res.	Res.	FOSR[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOSR[7:0]							
	reset value	0	0	0				0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0

Table 228. DFSDM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x398	DFSDM_ FLT5JDATAR	JDATA[23:0]																								Res.	Res.	Res.	Res.	Res.	JDATACH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x39C	DFSDM_ FLT5RDATAR	RDATA[23:0]																								Res.	Res.	Res.	RPEND	Res.	RDATA CH[2:0]		
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0		0	0	0
0x3A0	DFSDM_ FLT5AWHTR	AWHT[23:0]																								Res.	Res.	Res.	Res.	BKAWH[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x3A4	DFSDM_ FLT5AWLTR	AWLT[23:0]																								Res.	Res.	Res.	Res.	BKAWL[3:0]			
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x3A8	DFSDM_ FLT5AWSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWHTF[7:0]								AWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3AC	DFSDM_ FLT5AWCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLRAWHTF[7:0]								CLRAWLTF[7:0]							
	reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B0	DFSDM_ FLT5EXMAX	EXMAX[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMAXCH[2:0]		
	reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x3B4	DFSDM_ FLT5EXMIN	EXMIN[23:0]																								Res.	Res.	Res.	Res.	Res.	EXMINCH[2:0]		
	reset value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1						0	0	0
0x3B8	DFSDM_ FLT5CNVTIMR	CNVCNT[27:0]																												Res.	Res.	Res.	Res.
	reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x3BC - 0x4FC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x7F0	DFSDM_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBF[7:0]								NBT[7:0]							
	reset value																	4								8							
0x7F4	DFSDM_ VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	reset value																									0x				0x			
0x7F8	DFSDM_ IPIDR	ID[31:0]																															
	reset value	0x00110031																															
0x7FC	DFSDM_ SIDR	SID[31:0]																															
	reset value	0xA3C5DD02																															

Refer to *Section 2.5 on page 153* for the register boundary addresses.

33 Digital camera interface (DCMI)

33.1 Introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

33.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

33.3 DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

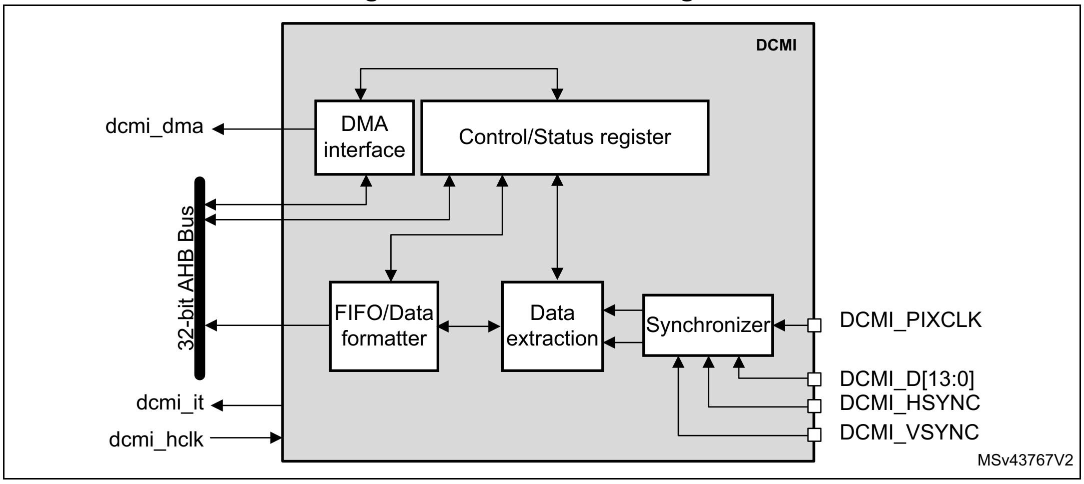
The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

33.3.1 DCMI block diagram

Figure 269 shows the DCMI block diagram.

Figure 269. DCMI block diagram



33.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

Table 229. DCMI input/output pins

Mode	Pin name	Signal type	Description
8 bits 10 bits 12 bits 14 bits	DCMI_D[7:0] DCMI_D[9:0] DCMI_D[11:0] DCMI_D[13:0]	Inputs	DCMI data
DCMI_PIXCLK		Input	Pixel clock
DCMI_HSYNC		Input	Horizontal synchronization / Data valid
DCMI_VSYNC		Input	Vertical synchronization

The following table shows DCMI internal signals.

Table 230. DCMI internal input/output signals

Internal signal name	Signal type	Description
dcmi_dma	Output	DCMI DMA request
dcmi_it	Output	DCMI interrupt request
dcmi_hclk	Input	DCMI interface clock

33.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

33.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

33.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

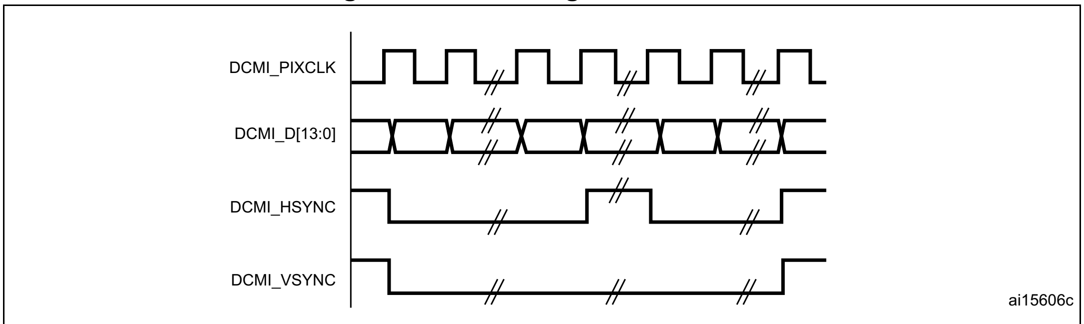
DCMI pins are shown in *Table 229*.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame

Figure 270. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.

Table 231. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 232. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 233. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 234. Positioning of captured data bytes in 32-bit words (14-bit width)

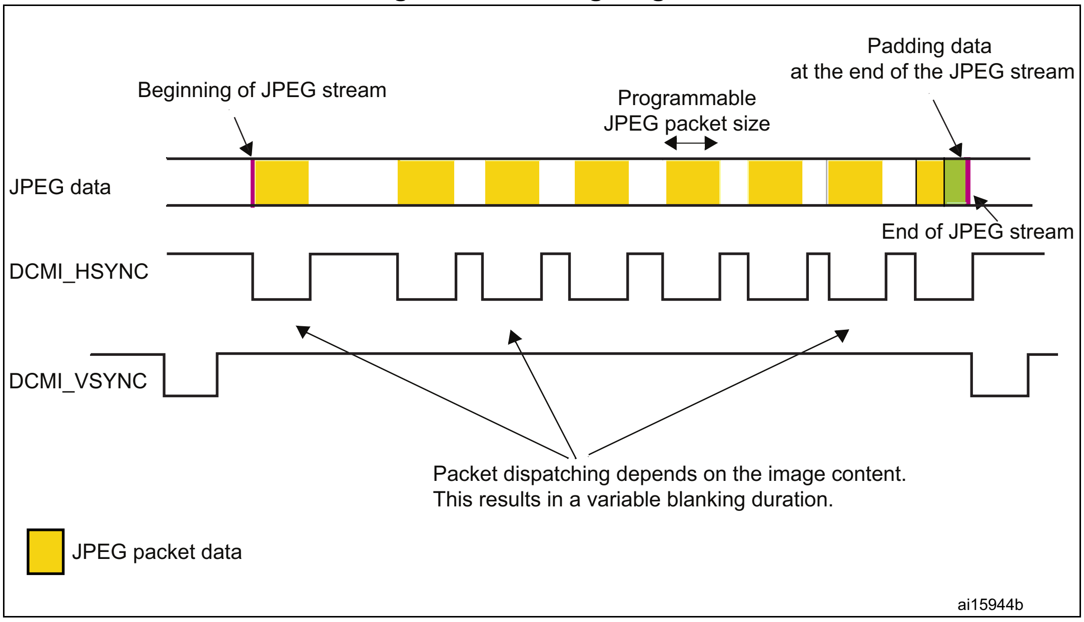
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

33.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. *Figure 271* shows the corresponding timing diagram.

Figure 271. Timing diagram



Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 33.5.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:
- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. A bit can therefore be selected to compare in the embedded code and detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

33.3.7 DCMI capture modes

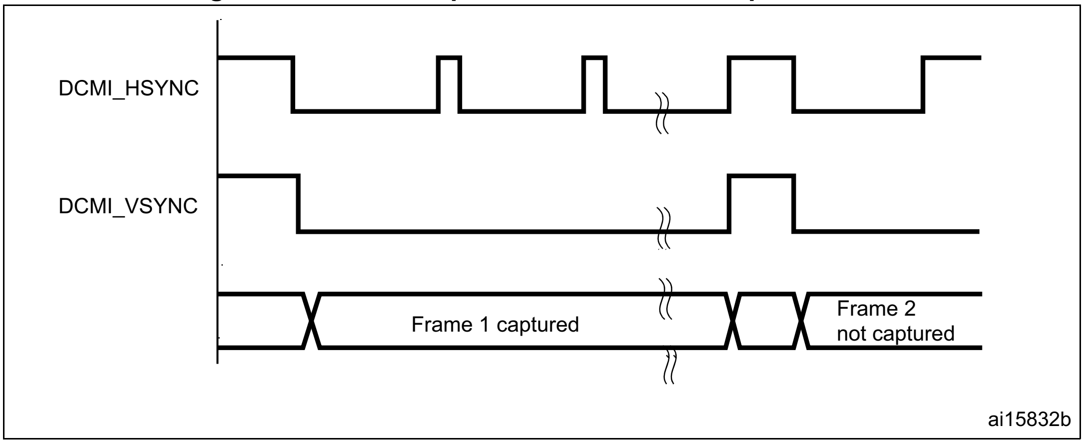
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 272. Frame capture waveforms in snapshot mode

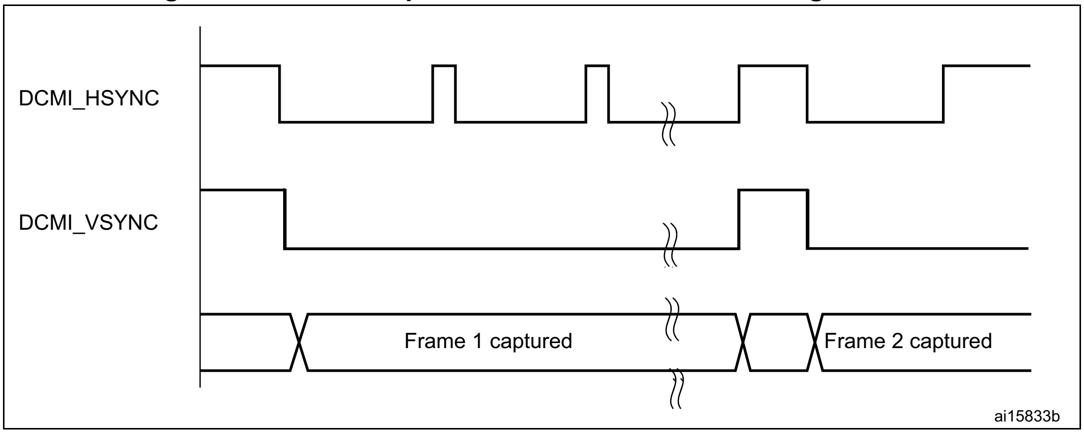


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 273. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

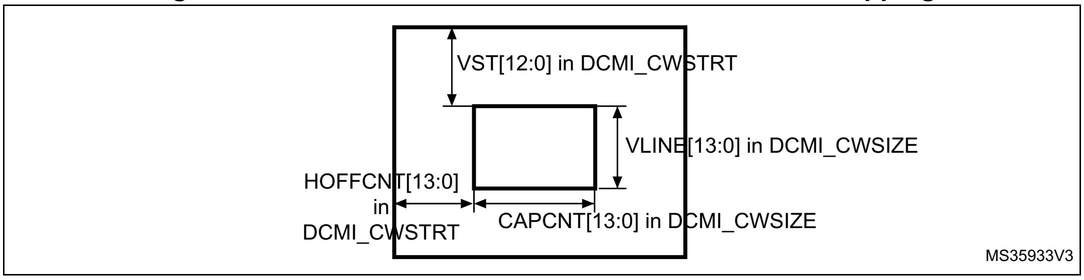
In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

33.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

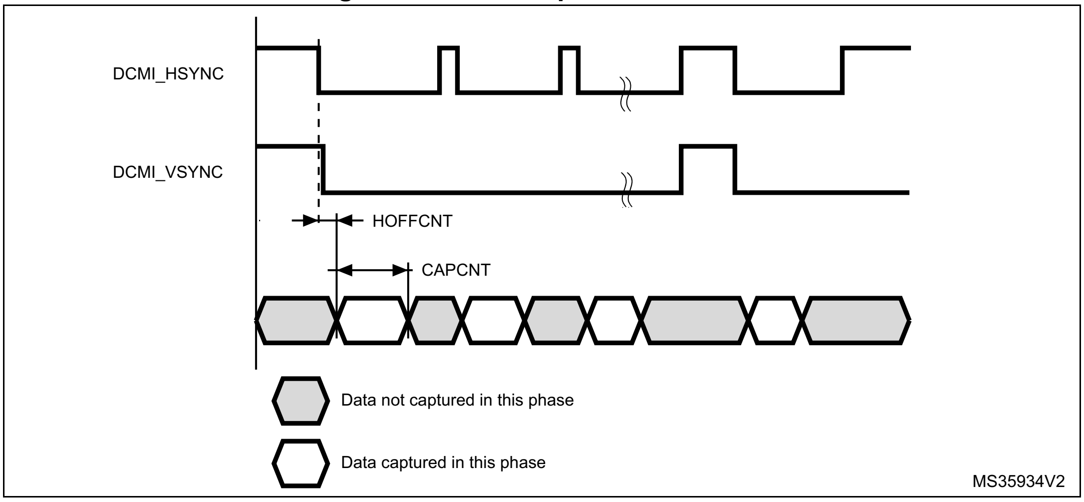
Figure 274. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 275. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

33.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

33.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

33.3.11 DCMI data format description

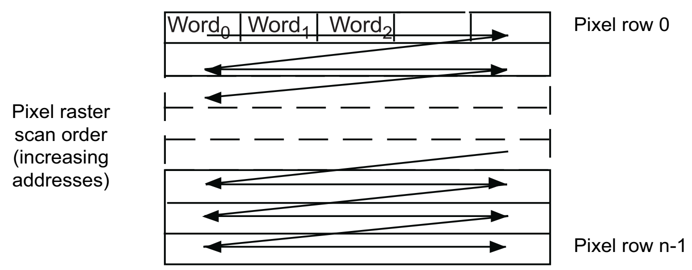
Data formats

The following types of data are supported:

- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred
- compressed data: JPEG

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

Figure 276. Pixel raster scan order



Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

Table 235. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGBRG)
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats.

Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

Table 236. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

Table 237. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in *Table 238*.

The result is a monochrome image having the same resolution as the original YCbCr data.

Table 238. Data storage in YCbCr progressive video format - Y extraction mode

Byte address	31:24	23:16	15:8	7:0
0	Y n + 3	Y n + 2	Y n + 1	Y n
4	Y n + 7	Y n + 6	Y n + 5	Y n + 4

Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

33.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

Table 239. DCMI interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exits Sleep mode	Exists Stop and Standby modes
dcmi_it	End of line	LINE_RIS	LINE_IE	Set LINE_ISC	Yes	No
	End of frame capture	FRAME_RIS	FRAME_IE	Set FRAME_ISC	Yes	No
	Overrun of data reception	OVR_RIS	OVR_IE	Set OVR_ISC	Yes	No
	Synchronization frame	VSYNC_RIS	VSYNC_IE	Set VSYNC_ISC	Yes	No
	Detection of an error in the embedded synchronization frame detection	ERR_RIS	ERR_IE	Set ERR_ISC	Yes	No

33.5 DCMI registers

Refer to *Section 1.2 on page 115* for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

33.5.1 DCMI control register (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OELS:** Odd/Even Line Select (Line Select Start)

This bit works in conjunction with the LSM field (LSM = 1).

0: Interface captures first line after the frame start, second one being dropped.

1: Interface captures second line from the frame start, first one being dropped.

Bit 19 **LSM:** Line Select mode

0: Interface captures all received lines.

1: Interface captures one line out of two.

Bit 18 **OEBS:** Odd/Even Byte Select (Byte Select Start)

This bit works in conjunction with BSM field (BSM ≠ 00).

0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.

1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

Bits 17:16 **BSM[1:0]:** Byte Select mode

00: Interface captures all received data.

01: Interface captures every other byte from the received data.

10: Interface captures one byte out of four.

11: Interface captures two bytes out of four.

Note: This mode only works for EDM[1:0] = 00. For all other EDM values, this field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** DCMI enable

0: DCMI disabled

1: DCMI enabled

Note: The DCMI configuration registers must be programmed correctly before enabling this bit.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every pixel clock.

01: Interface captures 10-bit data on every pixel clock.

10: Interface captures 12-bit data on every pixel clock.

11: Interface captures 14-bit data on every pixel clock.

Bits 9:8 **FCRC[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.

00: All frames are captured.

01: Every alternate frame captured (50% bandwidth reduction)

10: One frame out of four captured (75% bandwidth reduction)

11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity

This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.

0: DCMI_VSYNC active low

1: DCMI_VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity

This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.

0: DCMI_HSYNC active low
1: DCMI_HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity

This bit configures the capture edge of the pixel clock.

0: Falling edge active
1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select

0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.

Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.

This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format

0: Uncompressed video format
1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP:** Crop feature

0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.
1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.
1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled
1: Capture enabled

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the first frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit is effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.

33.5.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO.
1: FIFO contains valid data.
0: FIFO empty

Bit 1 **VSYNC**: Vertical synchronization

This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active frame
1: synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**: Horizontal synchronization

This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active line
1: synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

33.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_RIS**: Line raw interrupt status

This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.

Bit 3 **VSYNC_RIS**: DCMI_VSYNC raw interrupt status

This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.

Bit 2 **ERR_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS**: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.

Bit 0 **FRAME_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.

33.5.4 DCMI interrupt enable register (DCMI_IER)

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_IE**: Line interrupt enable
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received.

Bit 3 **VSYNC_IE**: DCMI_VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE**: Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE**: Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE**: Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

33.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_MIS**: Line masked interrupt status
This bit gives the status of the masked line interrupt.
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status
This bit gives the status of the masked VSYNC interrupt.
0: No interrupt is generated on DCMI_VSYNC transitions.
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status
This bit gives the status of the masked synchronization error interrupt.
0: No interrupt is generated on a synchronization error.
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status
This bit gives the status of the masked overflow interrupt.
0: No interrupt is generated on overrun.
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status
This bit gives the status of the masked capture complete interrupt
0: No interrupt is generated after a complete capture.
1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

33.5.6 DCMI interrupt clear register (DCMI_ICR)

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE _ISC	VSYNC _ISC	ERR _ISC	OVR _ISC	FRAME _ISC
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LINE_ISC**: line interrupt status clear
Setting this bit clears the LINE_RIS flag in the DCMI_RIS register.

Bit 3 **VSYNC_ISC**: Vertical Synchronization interrupt status clear
Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register.

Bit 2 **ERR_ISC**: Synchronization error interrupt status clear
Setting this bit clears the ERR_RIS flag in the DCMI_RIS register.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC**: Overrun interrupt status clear

Setting this bit clears the OVR_RIS flag in the DCMI_RIS register.

Bit 0 **FRAME_ISC**: Capture complete interrupt status clear

Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register.

33.5.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEC[7:0]								LEC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSC[7:0]								FSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEC[7:0]**: Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.

33.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FEU[7:0]								LEU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSU[7:0]								FSU[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask
This byte specifies the mask to be applied to the code of the frame end delimiter.
0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.
1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask
This byte specifies the mask to be applied to the code of the line end delimiter.
0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.
1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask
This byte specifies the mask to be applied to the code of the line start delimiter.
0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.
1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask
This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.

33.5.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	VST[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	HOFFCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **VST[12:0]**: Vertical start line count
The image capture starts with this line number. Previous line data are ignored.
0x0000: line 1
0x0001: line 2
0x0002: line 3
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count
This value gives the number of pixel clocks to count before starting a capture.

33.5.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	VLINE[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	CAPCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **VLINE[13:0]**: Vertical line count
This value gives the number of lines to be captured from the starting point.
0x0000: 1 line
0x0001: 2 lines
0x0002: 3 lines
....

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CAPCNT[13:0]**: Capture count
This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces.
0x0000: 1 pixel
0x0001: 2 pixels
0x0002: 3 pixels
....

33.5.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BYTE1[7:0]								BYTE0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

33.5.12 DCMI register map

Table 240. DCMI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]		Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value												0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DCMI_ESUR	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0]													Res.	Res.	HOFFCNT[13:0]													
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Res.	Res.	VLINE[13:0]														Res.	Res.	CAPCNT[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5* for the register boundary addresses.

34 LCD-TFT display controller (LTDC)

34.1 Introduction

The LCD-TFT (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (red, green, blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD and TFT panels.

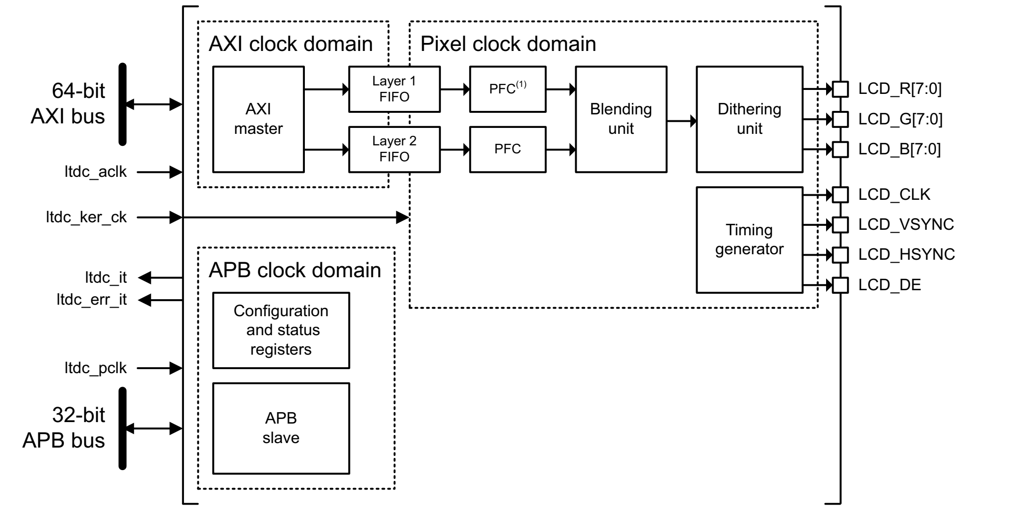
34.2 LTDC main features

- 24-bit RGB parallel pixel output; 8 bits-per-pixel (RGB888)
- 2 display layers with dedicated FIFO (64x64-bit)
- Color look-up table (CLUT) up to 256 color (256x24-bit) per layer
- Programmable timings for different display panels
- Programmable background color
- Programmable polarity for HSYNC, VSYNC and data enable
- Up to 8 input color formats selectable per layer:
  – ARGB8888
  – RGB888
  – RGB565
  – ARGB1555
  – ARGB4444
  – L8 (8-bit luminance or CLUT)
  – AL44 (4-bit alpha + 4-bit luminance)
  – AL88 (8-bit alpha + 8-bit luminance)
- Pseudo-random dithering output for low bits per channel
  – Dither width 2 bits for red, green, blue
- Flexible blending between two layers using alpha value (per pixel or constant)
- Color keying (transparency color)
- Programmable window position and size
- Supports thin film transistor (TFT) color displays
- AXI master interface with burst of 16 double-words
- Up to 4 programmable interrupt events

34.3 LTDC functional description

34.3.1 LTDC block diagram

Figure 277. LTDC block diagram



(1) PFC: pixel format converter, performing the pixel format conversion from the selected input pixel format of a layer to words.

34.3.2 LTDC pins and internal signals

The table below summarizes the LTDC signal interface.

Table 241. LTDC external pins

LCD-TFT signals	Signal type	Description
LCD_CLK	Output	Clock output
LCD_HSYNC	Output	Horizontal synchronization
LCD_VSYNC	Output	Vertical synchronization
LCD_DE	Output	Not data enable
LCD_R[7:0]	Output	8-bit Red data
LCD_G[7:0]	Output	8-bit Green data
LCD_B[7:0]	Output	8-bit Blue data

The LTDC pins must be configured by the user application. The unused pins can be used for other purposes.

For LTDC outputs up to 24 bits (RGB888), if less than 8 bpp are used to output for example RGB565 or RGB666 to interface on 16- or 18-bit displays, the RGB display data lines must be connected to the MSB of the LTDC RGB data lines.

As an example, in the case of an LTDC interfacing with a RGB565 16-bit display, the LTDC display R[4:0], G[5:0] and B[4:0] data lines pins must be connected to the LCD_R[7:3], LCD_G[7:2] and LCD_B[7:3] pins.

The internal signals of the LTDC are given in the table below.

Table 242. LTDC internal signals

Names	Signal type	Description
ltdc_aclk	Input	LTDC AXI clock
ltdc_pclk	Input	LTDC APB clock for register access
ltdc_ker_ck	Input	LTDC kernel clock used for LCD_CLK (pixel clock) generation
ltdc_it	Output	LTDC global interrupt request
ltdc_err_it	Output	LTDC global error interrupt request

34.3.3 LTDC reset and clocks

The LTDC controller peripheral uses the following clock domains:

- AXI clock domain (ltdc_aclk)
  This domain contains the LTDC AXI master interface for data transfer from the memories to the layer FIFO and the frame-buffer configuration register.
- APB clock domain (ltdc_pclk)
  This domain contains the global configuration registers and the interrupt register.
- Pixel clock domain (LCD_CLK)
  This domain contains the pixel data generation, the layer configuration register as well as the LTDC interface signal generator. The LCD_CLK output must be configured following the panel requirements. The LCD_CLK is generated from a specific PLL output (refer to the reset and clock control section).

The table below summarizes the clock domain for each register.

Table 243. Clock domain for each register

LTDC register	Clock domain
LTDC_LxCR	ltdc_aclk
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_SRCR	ltdc_pclk
LTDC_IER	
LTDC_ISR	
LTDC_ICR	

Table 243. Clock domain for each register (continued)

LTDC register	Clock domain
LTDC_SSCR	Pixel clock (LCD_CLK)
LTDC_BPCR	
LTDC_AWCR	
LTDC_TWCR	
LTDC_GCR	
LTDC_BCCR	
LTDC_LIPCR	
LTDC_CPSR	
LTDC_CDSR	
LTDC_LxWHPCR	
LTDC_LxWVPCR	
LTDC_LxCKCR	
LTDC_LxPFCR	
LTDC_LxCACR	
LTDC_LxDCCR	
LTDC_LxBFCR	
LTDC_LxCLUTWR	

Care must be taken while accessing the LTDC registers, the APB bus is stalled during the access for a given time period (see the table below).

Table 244. LTDC register access and update durations

	Register clock domain		
	AXI domain	APB domain	Pixel clock domain
Register read access duration	7 x ltdc_pclk + 5 x ltdc_aclk	7 x ltdc_pclk	7 x ltdc_pclk + 5 x ltdc_ker_clk
Register write access duration	6 x ltdc_pclk + 5 x ltdc_aclk	6 x ltdc_pclk	6 x ltdc_pclk + 5 x ltdc_ker_clk

The LTDC controller can be reset by setting the corresponding bit in the RCC. It resets the three clock domains.

34.4 LTDC programmable parameters

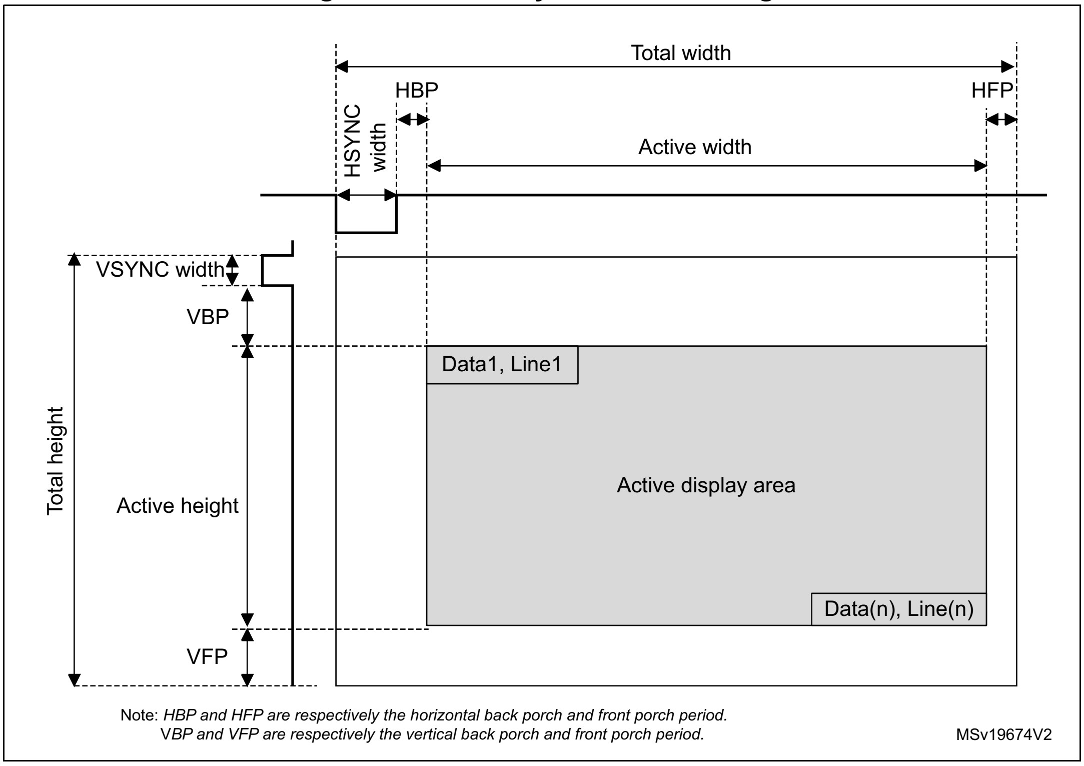
The LTDC controller provides flexible configurable parameters. It can be enabled or disabled through the LTDC_GCR register.

34.4.1 LTDC global configuration parameters

Synchronous timings

The figure below presents the configurable timing parameters generated by the synchronous timings generator block presented in the block diagram *Figure 277*. It generates the horizontal and vertical synchronization timings panel signals, the pixel clock and the data enable signals.

Figure 278. LTDC synchronous timings



The LTDC programmable synchronous timings are the following:

- HSYNC and VSYNC width: horizontal and vertical synchronization width, configured by programming a value of HSYNC width - 1 and VSYNC width - 1 in the LTDC_SSCR register
- HBP and VBP: horizontal and vertical synchronization back porch width, configured by programming the accumulated value HSYNC width + HBP - 1 and the accumulated value VSYNC width + VBP - 1 in the LTDC_BPCR register.
- Active width and active height: the active width and active height are configured by programming the accumulated value HSYNC width + HBP + active width - 1 and the

accumulated value VSYNC width + VBP + active height - 1 in the LTDC_AWCR register.

- Total width: the total width is configured by programming the accumulated value HSYNC width + HBP + active width + HFP - 1 in the LTDC_TWCR register. The HFP is the horizontal front porch period.
- Total height: the total height is configured by programming the accumulated value VSYNC height + VBP + active height + VFP - 1 in the LTDC_TWCR register. The VFP is the vertical front porch period.

Note: When the LTDC is enabled, the timings generated start with X/Y = 0/0 position as the first horizontal synchronization pixel in the vertical synchronization area and following the back porch, active data display area and the front porch.
When the LTDC is disabled, the timing generator block is reset to X = total width - 1, Y = total height - 1 and held the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore only blanking data is output continuously.

Example of synchronous timings configuration

LTDC timings (must be extracted from panel datasheet):

- horizontal and vertical synchronization width: 0xA pixels and 0x2 lines
- horizontal and vertical back porch: 0x14 pixels and 0x2 lines
- active width and active height: 0x140 pixels, 0xF0 lines (320x240)
- horizontal front porch: 0xA pixels
- vertical front porch: 0x4 lines

The programmed values in the LTDC timings registers are:

- LTDC_SSCR register to be programmed to 0x00090001
(HSW[11:0] is 0x9 and VSH[11:0] is 0x1)
- LTDC_BPCR register to be programmed to 0x001D0003
(AHBP[11:0] is 0x1D (0xA+ 0x13) and AVBP[11:0] is 0x3 (0x2 + 0x1))
- LTDC_AWCR register to be programmed to 0x015D00F3
(AAW[11:0] is 0x15D (0xA +0x14 +0x13F) and AAH[11:0] is 0xF3 (0x2 + 0x2 + 0xEF))
- LTDC_TWCR register to be programmed to 0x00000167
(TOTALW[11:0] is 0x167 (0xA +0x14 +0x140 + 0x9))
- LTDC_THCR register to be programmed to 0x000000F7
(TOTALH[11:0] is 0xF7 (0x2 +0x2 + 0xF0 + 3))

Programmable polarity

The horizontal and vertical synchronization, data enable and pixel clock output signals polarity can be programmed to active high or active low through the LTDC_GCR register.

Background color

A constant background color (RGB888) can programmed through the LTDC_BCCR register. It is used for blending with the bottom layer.

Dithering

The dithering pseudo-random technique using an LFSR is used to add a small random value (threshold) to each pixel color channel (R, G or B) value, thus rounding up the MSB in some cases when displaying a 24-bit data on 18-bit display. Thus the dithering technique is used to round data which is different from one frame to the other.

The dithering pseudo-random technique is the same as comparing LSBs against a threshold value and adding a 1 to the MSB part only, if the LSB part is ≥ the threshold. The LSBs are typically dropped once dithering was applied.

The width of the added pseudo-random value is two bits for each color channel: two bits for red, two bits for green and two bits for blue.

Once the LTDC is enabled, the LFSR starts running with the first active pixel and it is kept running even during blanking periods and when dithering is switched off. If the LTDC is disabled, the LFSR is reset.

The dithering can be switched on and off on the fly through the LTDC_GCR register.

Reload shadow registers

Some configuration registers are shadowed. The shadow registers values can be reloaded immediately to the active registers when writing to these registers or at the beginning of the vertical blanking period following the configuration in the LTDC_SRCR register. If the immediate reload configuration is selected, the reload must be activated only when all new registers have been written.

The shadow registers must not be modified again before the reload is done. Reading from the shadow registers returns the actual active value. The new written value can only be read after the reload has taken place.

A register reload interrupt can be generated if enabled in the LTDC_IER register.

The shadowed registers are all Layer1 and Layer2 registers except LTDC_LxCLUTWR.

Interrupt generation event

Refer to *Section 34.5: LTDC interrupts* for the interrupt configuration.

34.4.2 Layer programmable parameters

Up to two layers can be enabled, disabled and configured separately. The layer display order is fixed and it is bottom up. If two layers are enabled, the layer2 is the top displayed window.

Windowing

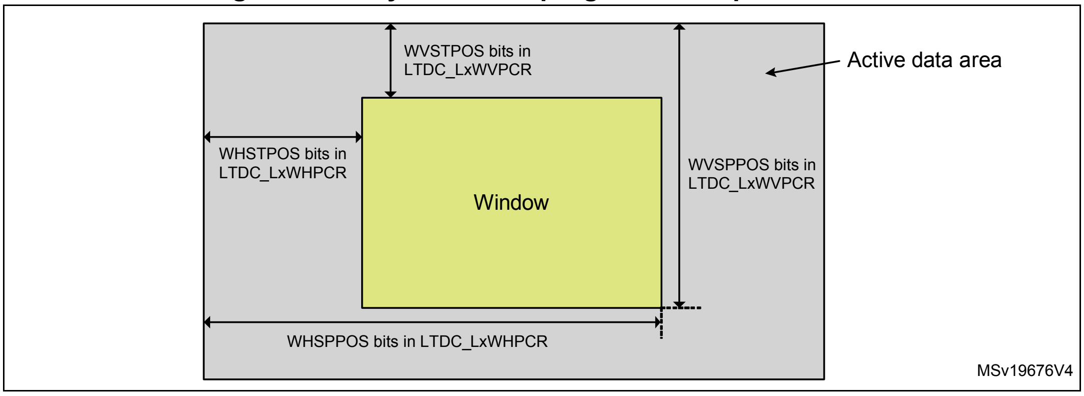
Every layer can be positioned and resized and it must be inside the active display area.

The window position and size are configured through the top-left and bottom-right X/Y positions and the internal timing generator that includes the synchronous, back porch size and the active data area. Refer to LTDC_LxWHPCR and LTDC_WVPCR registers.

The programmable layer position and size defines the first/last visible pixel of a line and the first/last visible line in the window. It allows to display either the full image frame or only a part of the image frame (see *Figure 279*):

- The first and the last visible pixel in the layer are set by configuring the WHSTPOS[11:0] and WHSPPOS[11:0] in the LTDC_LxWHPCR register.
- The first and the last visible lines in the layer are set by configuring the WVSTPOS[11:0] and WVSPPOS[11:0] in the LTDC_LxWVPCR register.

Figure 279. Layer window programmable parameters



Pixel input format

The programmable pixel format is used for the data stored in the frame buffer of a layer.

Up to eight input pixel formats can be configured for every layer through the LTDC_LxPFCR register

The pixel data is read from the frame buffer and then transformed to the internal 8888 (ARGB) format as follows: components having a width of less than 8 bits get expanded to 8 bits by bit replication. The selected bit range is concatenated multiple times until it is longer than 8 bits. Of the resulting vector, the 8 MSB bits are chosen. Example: 5 bits of an RGB565 red channel become (bit positions) 43210432 (the three LSBs are filled with the three MSBs of the five bits)

The table below describes the pixel data mapping depending on the selected format.

Table 245. Pixel data mapping versus color format

ARGB8888			
@+3 A_x[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 A_{x+1}[7:0]	@+6 R_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 B_{x+1}[7:0]
RGB888			
@+3 B_{x+1}[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 G_{x+2}[7:0]	@+6 B_{x+2}[7:0]	@+5 R_{x+1}[7:0]	@+4 G_{x+1}[7:0]
RGB565			
@+3 R_{x+1}[4:0] G_{x+1}[5:3]	@+2 G_{x+1}[2:0] B_{x+1}[4:0]	@+1 R_x[4:0] G_x[5:3]	@ G_x[2:0] B_x[4:0]
@+7 R_{x+3}[4:0] G_{x+3}[5:3]	@+6 G_{x+3}[2:0] B_{x+3}[4:0]	@+5 R_{x+2}[4:0] G_{x+2}[5:3]	@+4 G_{x+2}[2:0] B_{x+2}[4:0]
ARGB1555			

Table 245. Pixel data mapping versus color format (continued)

@+3 $A_{x+1}[0]R_{x+1}[4:0]$ $G_{x+1}[4:3]$	@+2 $G_{x+1}[2:0]$ $B_{x+1}[4:0]$	@+1 $A_x[0]$ $R_x[4:0]$ $G_x[4:3]$	@ $G_x[2:0]$ $B_x[4:0]$
@+7 $A_{x+3}[0]R_{x+3}[4:0]$ $G_{x+3}[4:3]$	@+6 $G_{x+3}[2:0]$ $B_{x+3}[4:0]$	@+5 $A_{x+2}[0]R_{x+2}[4:0]G_{x+2}[4:3]$	@+4 $G_{x+2}[2:0]$ $B_{x+2}[4:0]$
ARGB4444			
@+3 $A_{x+1}[3:0]R_{x+1}[3:0]$	@+2 $G_{x+1}[3:0]$ $B_{x+1}[3:0]$	@+1 $A_x[3:0]$ $R_x[3:0]$	@ $G_x[3:0]$ $B_x[3:0]$
@+7 $A_{x+3}[3:0]R_{x+3}[3:0]$	@+6 $G_{x+3}[3:0]$ $B_{x+3}[3:0]$	@+5 $A_{x+2}[3:0]R_{x+2}[3:0]$	@+4 $G_{x+2}[3:0]$ $B_{x+2}[3:0]$
L8			
@+3 $L_{x+3}[7:0]$	@+2 $L_{x+2}[7:0]$	@+1 $L_{x+1}[7:0]$	@ $L_x[7:0]$
@+7 $L_{x+7}[7:0]$	@+6 $L_{x+6}[7:0]$	@+5 $L_{x+5}[7:0]$	@+4 $L_{x+4}[7:0]$
AL44			
@+3 $A_{x+3}[3:0]$ $L_{x+3}[3:0]$	@+2 $A_{x+2}[3:0]$ $L_{x+2}[3:0]$	@+1 $A_{x+1}[3:0]$ $L_{x+1}[3:0]$	@ $A_x[3:0]$ $L_x[3:0]$
@+7 $A_{x+7}[3:0]$ $L_{x+7}[3:0]$	@+6 $A_{x+6}[3:0]$ $L_{x+6}[3:0]$	@+5 $A_{x+5}[3:0]$ $L_{x+5}[3:0]$	@+4 $A_{x+4}[3:0]$ $L_{x+4}[3:0]$
AL88			
@+3 $A_{x+1}[7:0]$	@+2 $L_{x+1}[7:0]$	@+1 $A_x[7:0]$	@ $L_x[7:0]$
@+7 $A_{x+3}[7:0]$	@+6 $L_{x+3}[7:0]$	@+5 $A_{x+2}[7:0]$	@+4 $L_{x+2}[7:0]$

Color look-up table (CLUT)

The CLUT can be enabled at run-time for every layer through the LTDC_LxCR register and it is only useful in case of indexed color when using the L8, AL44 and AL88 input pixel format.

First, the CLUT must be loaded with the R, G and B values that replace the original R, G, B values of that pixel (indexed color). Each color (RGB value) has its own address that is the position within the CLUT.

The R, G and B values and their own respective address are programmed through the LTDC_LxCLUTWR register:

- In case of L8 and AL88 input pixel format, the CLUT must be loaded by 256 colors. The address of each color is configured in the CLUTADD bits in the LTDC_LxCLUTWR register.

- In case of AL44 input pixel format, the CLUT must be loaded by only16 colors. The address of each color must be filled by replicating the 4-bit L channel to 8-bit as follows:
  - L0 (indexed color 0), at address 0x00
  - L1, at address 0x11
  - L2, at address 0x22
  - .....
  - L15, at address 0xFF

Color frame buffer address

Every layer has a start address for the color frame buffer configured through the LTDC_LxCFBAR register.

When a layer is enabled, the data is fetched from the color frame buffer.

Color frame buffer length

Every layer has a total line length setting for the color frame buffer in bytes and a number of lines in the frame buffer configurable in the LTDC_LxCFBLR and LTDC_LxCFBLNR register respectively.

The line length and the number of lines settings are used to stop the prefetching of data to the layer FIFO at the end of the frame buffer:

- If it is set to less bytes than required, a FIFO underrun interrupt is generated if it has been previously enabled.
- If it is set to more bytes than actually required, the useless data read from the FIFO is discarded. The useless data is not displayed.

Color frame buffer pitch

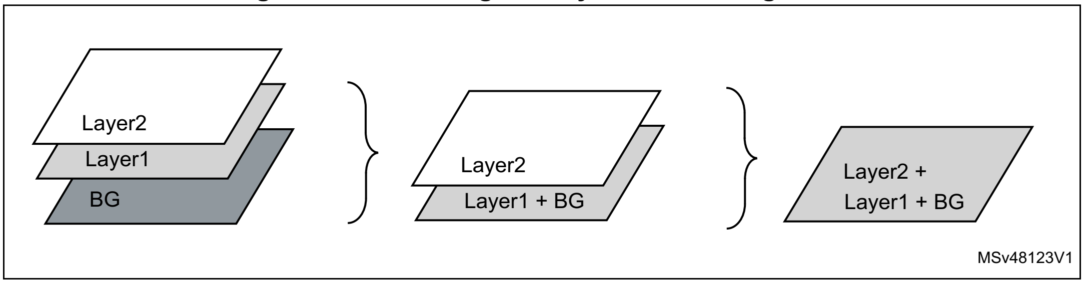
Every layer has a configurable pitch for the color frame buffer, that is the distance between the start of one line and the beginning of the next line in bytes. It is configured through the LTDC_LxCFBLR register.

Layer blending

The blending is always active and the two layers can be blended following the blending factors configured through the LTDC_LxBFCR register.

The blending order is fixed and it is bottom up. If two layers are enabled, first the Layer1 is blended with the Background color, then the layer2 is blended with the result of blended color of layer1 and the background. Refer to the figure below.

Figure 280. Blending two layers with background



Default color

Every layer can have a default color in the format ARGB which is used outside the defined layer window or when a layer is disabled.

The default color is configured through the LTDC_LxDCCR register.

The blending is always performed between the two layers even when a layer is disabled. To avoid displaying the default color when a layer is disabled, keep the blending factors of this layer in the LTDC_LxBFCR register to their reset value.

Color keying

A color key (RGB) can be configured to be representative for a transparent pixel.

If the color keying is enabled, the current pixels (after format conversion and before CLUT respectively blending) are compared to the color key. If they match for the programmed RGB value, all channels (ARGB) of that pixel are set to 0.

The color key value can be configured and used at run-time to replace the pixel RGB value.

The color keying is enabled through the LTDC_LxCKCR register.

The color keying is configured through the LTDC_LxCKCR register. The programmed value depends on the pixel format as it is compared to current pixel after pixel format conversion to ARGB888.

Example: if the a mid-yellow color (50 % red + 50 % green) is used as the transparent color key:

- In RGB565, the mid-yellow color is 0x8400. Set the LTDC_LxCKCR to 0x848200.
- In ARGB8888, the mid-yellow color is 0x808000. Set LTDC_LxCKCR to 0x808000.
- In all CLUT-based color modes (L8, AL88, AL44), set one of the palette entry to the mid-yellow color 0x808000 and set the LTDC_LxCKCR to 0x808000.

34.5 LTDC interrupts

The LTDC provides four maskable interrupts logically ORed to two interrupt vectors.

The interrupt sources can be enabled or disabled separately through the LTDC_IER register. Setting the appropriate mask bit to 1 enables the corresponding interrupt.

The two interrupts are generated on the following events:

- Line interrupt: generated when a programmed line is reached. The line interrupt position is programmed in the LTDC_LIPCR register
- Register reload interrupt: generated when the shadow registers reload is performed during the vertical blanking period
- FIFO underrun interrupt: generated when a pixel is requested from an empty layer FIFO
- Transfer error interrupt: generated when an AXI bus error occurs during data transfer

These interrupt events are connected to the NVIC controller as described in the figure below.

Figure 281. Interrupt events

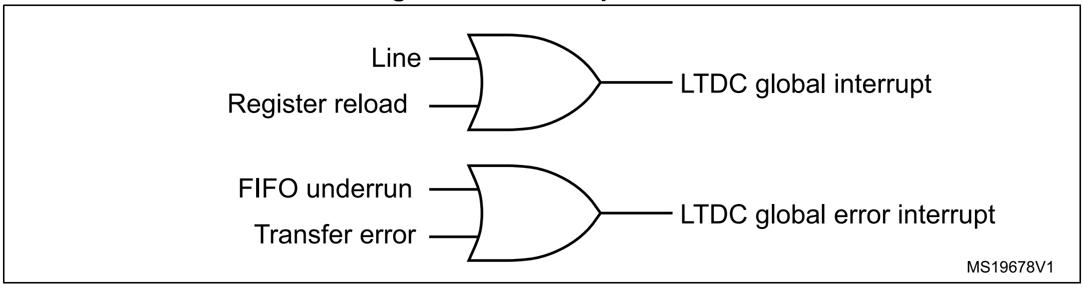


Table 246. LTDC interrupt requests

Interrupt event	Event flag	Enable control bit
Line	LIF	LIE
Register reload	RRIF	RRIEN
FIFO underrun	FUDERRIF	FUDERRIE
Transfer error	TERRIF	TERRIE

34.6 LTDC programming procedure

The steps listed below are needed to program the LTDC:

1. Enable the LTDC clock in the RCC register.
2. Configure the required pixel clock following the panel datasheet.
3. Configure the synchronous timings: VSYNC, HSYNC, vertical and horizontal back porch, active data area and the front porch timings following the panel datasheet as described in the *Section 34.4.1*.
4. Configure the synchronous signals and clock polarity in the LTDC_GCR register.
5. If needed, configure the background color in the LTDC_BCCR register.
6. Configure the needed interrupts in the LTDC_IER and LTDC_LIPCR register.
7. Configure the layer1/2 parameters by:
   – programming the layer window horizontal and vertical position in the LTDC_LxWHPCR and LTDC_WVPCR registers. The layer window must be in the active data area.
   – programming the pixel input format in the LTDC_LxPFCR register
   – programming the color frame buffer start address in the LTDC_LxCFBAR register
   – programming the line length and pitch of the color frame buffer in the LTDC_LxCFBLR register
   – programming the number of lines of the color frame buffer in the LTDC_LxCFBLNR register
   – if needed, loading the CLUT with the RGB values and its address in the LTDC_LxCLUTWR register
   – If needed, configuring the default color and the blending factors respectively in the LTDC_LxDCCR and LTDC_LxBFCR registers

8. Enable layer1/2 and if needed the CLUT in the LTDC_LxCR register.
9. If needed, enable dithering and color keying respectively in the LTDC_GCR and LTDC_LxCKCR registers. They can be also enabled on the fly.
10. Reload the shadow registers to active register through the LTDC_SRCR register.
11. Enable the LTDC controller in the LTDC_GCR register.
12. All layer parameters can be modified on the fly except the CLUT. The new configuration must be either reloaded immediately or during vertical blanking period by configuring the LTDC_SRCR register.

Note: All layer's registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. Thus, a new write to the same register overrides the previous configuration if not yet reloaded.

34.7 LTDC registers

34.7.1 LTDC identification register (LTDC_IDR)

Address offset: 0x000

Reset value: 0x0001 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER[7:0]							
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MINVER[7:0]								REV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **MAJVER[7:0]:** LTDC major version

Bits 15:8 **MINVER[7:0]:** LTDC minor version

Bits 7:0 **REV[7:0]:** revision

34.7.2 LTDC layer count register (LTDC_LCR)

Address offset: 0x004

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNBR[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LNBR[7:0]**: Number of layers

34.7.3 LTDC synchronization size configuration register (LTDC_SSCR)

Address offset: 0x008

Reset value: 0x0000 0000

This register defines the number of horizontal synchronization pixels minus 1 and the number of vertical synchronization lines minus 1. Refer to *Figure 278* and *Section 34.4* for an example of configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HSW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	VSH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HSW[11:0]**: Horizontal synchronization width (in units of pixel clock period)
This bitfield defines the number of Horizontal Synchronization pixel minus 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **VSH[11:0]**: Vertical synchronization height (in units of horizontal scan line)
This bitfield defines the vertical Synchronization height minus 1. It represents the number of horizontal synchronization lines.

34.7.4 LTDC back porch configuration register (LTDC_BPCR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (HSYNC width + HBP - 1) and the accumulated number of vertical synchronization and back porch lines minus 1 (VSYNC height + VBP - 1).
Refer to *Figure 278* and *Section 34.4* for an example of configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	AVBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AHBP[11:0]**: Accumulated horizontal back porch (in units of pixel clock period)
These bits defines the accumulated horizontal back porch width that includes the horizontal synchronization and horizontal back porch pixels minus 1.
The horizontal back porch is the period between horizontal synchronization going inactive and the start of the active display part of the next scan line.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **AVBP[11:0]:** Accumulated Vertical back porch (in units of horizontal scan line)
These bits define the accumulated vertical back porch width that includes the vertical synchronization and vertical back porch lines minus 1.
The vertical back porch is the number of horizontal scan lines at a start of frame to the start of the first active scan line of the next frame.

34.7.5 LTDC active width configuration register (LTDC_AWCR)

Address offset: 0x010

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (HSYNC width + HBP + active width - 1) and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (VSYNC height + VBP + active height - 1). Refer to *Figure 278* and *Section 34.4* for an example of configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AAW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	AAH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AAW[11:0]**: Accumulated active width (in units of pixel clock period)
These bits define the accumulated active width which includes the horizontal synchronization, horizontal back porch and active pixels minus 1.
The active width is the number of pixels in active display area of the panel scan line.
Refer to device datasheet for maximum active width supported following maximum pixel clock.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **AAH[11:0]**: Accumulated active height (in units of horizontal scan line)
These bits define the accumulated height which includes the vertical synchronization, vertical back porch and the active height lines minus 1. The active height is the number of active lines in the panel. Refer to the datasheet for maximum active height supported following maximum pixel clock.

34.7.6 LTDC total width configuration register (LTDC_TWCR)

Address offset: 0x014

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (HSYNC width + HBP + active width + HFP - 1) and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (VSYNC height + VBP + active height + VFP - 1). Refer to *Figure 278* and *Section 34.4* for an example of configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TOTALW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TOTALH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **TOTALW[11:0]**: Total width (in units of pixel clock period)
These bits define the accumulated total width which includes the horizontal synchronization, horizontal back porch, active width and horizontal front porch pixels minus 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **TOTALH[11:0]**: Total height (in units of horizontal scan line)
These bits define the accumulated height which includes the vertical synchronization, vertical back porch, the active height and vertical front porch height lines minus 1.

34.7.7 LTDC global control register (LTDC_GCR)

Address offset: 0x018

Reset value: 0x0000 2220

This register defines the global configuration of the LCD-TFT controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN
rw	rw	rw	rw												rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	Res.	LTDCEN
	r	r	r		r	r	r		r	r	r				rw

Bit 31 **HSPOL**: Horizontal synchronization polarity
This bit is set and cleared by software.
0: Horizontal synchronization polarity is active low.
1: Horizontal synchronization polarity is active high.

Bit 30 **VSPOL**: Vertical synchronization polarity
This bit is set and cleared by software.
0: Vertical synchronization is active low.
1: Vertical synchronization is active high.

Bit 29 **DEPOL**: Not data enable polarity
This bit is set and cleared by software.
0: Not data enable polarity is active low.
1: Not data enable polarity is active high.

Bit 28 **PCPOL**: Pixel clock polarity
This bit is set and cleared by software.
0: Pixel clock polarity is active low.
1: Pixel clock is active high.

Bits 27:17 Reserved, must be kept at reset value.

Bit 16 **DEN**: Dither enable
This bit is set and cleared by software.
0: Dither disabled
1: Dither enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DRW[2:0]**: Dither red width
These bits return the dither red bits.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DGW[2:0]**: Dither green width
These bits return the dither green bits.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DBW[2:0]**: Dither blue width
These bits return the dither blue bits.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LCD-TFT controller enable
This bit is set and cleared by software.
0: LTDC disabled
1: LTDC enabled

34.7.8 LTDC global configuration 1 register (LTDC_GC1R)

Address offset: 0x01C

Reset value: 0x6BE2 D888

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BMEN	Res.	STREN	DWP	SPP	IPP	TP	LNIP	BBEN	BCP	SHREN	Res.	GCT[2:0]			Res.
r		r	r	r	r	r	r	r	r	r		r	r	r	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DT[1:0]		Res.	PRBEN	WRCH[3:0]				WGCH[3:0]				WBCH[3:0]			
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **BMEN**: Blind mode enabled

Bit 30 Reserved, must be kept at reset value.

Bit 29 **STREN**: Status register enabled

Bit 28 **DWP:** Dither width programmable

Bit 27 **SPP**: Sync polarity programmable

Bit 26 **IPP**: IRQ polarity programmable

Bit 25 **TP**: Timing programmable

Bit 24 **LNIP**: Line IRQ position

Bit 23 **BBEN**: Background blending enabled

Bit 22 **BCP**: Background color programmable

Bit 21 **SHREN**: Shadow registers enabled

Bit 20 Reserved, must be kept at reset value.

Bits 19:17 **GCT[2:0]**: Gamma correction technique

Bit 16 Reserved, must be kept at reset value.

Bits 15:14 **DT[1:0]**: Dithering technique

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PRBEN**: Precise blending enabled

Bits 11:8 **WRCH[3:0]**: Width of red channel output

Bits 7:4 **WGCH[3:0]**: Width of green channel output

Bits 3:0 **WBCH[3:0]**: Width of blue channel output

34.7.9 LTDC global configuration 2 register (LTDC_GC2R)

Address offset: 0x020

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDCA	BW[2:0]			DPAEN	DVAEN	STSAEN	EDCEN
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **EDCA**: External display control ability

Bits 6:4 **BW[2:0]**: Bus width (log2 of number of bytes)

Bit 3 **DPAEN:** Secondary RGB output port enabled

Bit 2 **DVAEN:** Dual-view ability enabled

Bit 1 **STSAEN:** Slave timings synchronization ability enabled

Bit 0 **EDCEN:** Background layer ability enabled

34.7.10 LTDC shadow reload configuration register (LTDC_SRCR)

Address offset: 0x024

Reset value: 0x0000 0000

This register allows to reload either immediately or during the vertical blanking period, the shadow registers values to the active registers. The shadow registers are all Layer1 and Layer2 registers except the LTDC_L1CLUTWR and the LTDC_L2CLUTWR.

The shadow registers read back the active values. Until the reload has been done, the 'old' value is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBR**: Vertical blanking reload

This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).
0: No effect
1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: Immediate reload

This bit is set by software and cleared only by hardware after reload.
0: No effect
1: The shadow registers are reloaded immediately.

34.7.11 LTDC background color configuration register (LTDC_BCCR)

Address offset: 0x02C

Reset value: 0x0000 0000

This register defines the background color (RGB888).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCGREEN[7:0]								BCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BCRED[7:0]**: Background color red value

These bits configure the background red value.

Bits 15:8 **BCGREEN[7:0]**: Background color green value
These bits configure the background green value.

Bits 7:0 **BCBLUE[7:0]**: Background color blue value
These bits configure the background blue value.

34.7.12 LTDC interrupt enable register (LTDC_IER)

Address offset: 0x034

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIE	TERRIE	FUIE	LIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RRIE**: Register reload interrupt enable
This bit is set and cleared by software.
0: Register reload interrupt disable
1: Register reload interrupt enable

Bit 2 **TERRIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: Transfer error interrupt disable
1: Transfer error interrupt enable

Bit 1 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disable
1: FIFO underrun Interrupt enable

Bit 0 **LIE**: Line interrupt enable
This bit is set and cleared by software.
0: Line interrupt disable
1: Line interrupt enable

34.7.13 LTDC interrupt status register (LTDC_ISR)

Address offset: 0x038

Reset value: 0x0000 0000

This register returns the interrupt status flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIF	TERRIF	FUIF	LIF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RRIF**: Register reload interrupt flag

0: No register reload interrupt generated

1: Register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: Transfer error interrupt flag

0: No transfer error interrupt generated

1: Transfer error interrupt generated when a bus error occurs

Bit 1 **FUIF**: FIFO underrun interrupt flag

0: No FIFO underrun interrupt generated

1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and pixel data is read from the FIFO

Bit 0 **LIF**: Line interrupt flag

0: No line interrupt generated

1: Line interrupt generated when a programmed line is reached

34.7.14 LTDC interrupt clear register (LTDC_ICR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRRIF	CTERRIF	CFUIF	CLIF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CRRIF**: Clear register reload interrupt flag

0: No effect

1: Clear the RRIF flag in the LTDC_ISR register

Bit 2 **CTERRIF**: Clear the transfer error interrupt flag
0: No effect
1: Clear the TERRIF flag in the LTDC_ISR register.

Bit 1 **CFUIF**: Clear the FIFO underrun interrupt flag
0: No effect
1: Clear the FUDERRIF flag in the LTDC_ISR register.

Bit 0 **CLIF**: Clear the line interrupt flag
0: No effect
1: Clear the LIF flag in the LTDC_ISR register.

34.7.15 LTDC line interrupt position configuration register (LTDC_LIPCR)

Address offset: 0x040

Reset value: 0x0000 0000

This register defines the position of the line interrupt. The line value to be programmed depends on the timings parameters. Refer to *Figure 278*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	LIPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LIPOS[11:0]**: Line interrupt position
These bits configure the line interrupt position.

34.7.16 LTDC current position status register (LTDC_CPSR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CXPOS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CYPOS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CXPOS[15:0]**: Current X position
These bits return the current X position.

Bits 15:0 **CYPOS[15:0]**: Current Y position
These bits return the current Y position.

34.7.17 LTDC current display status register (LTDC_CDSR)

Address offset: 0x048

Reset value: 0x0000 000F

This register returns the status of the current display phase which is controlled by the HSYNC, VSYNC, and horizontal/vertical DE signals.

Example: if the current display phase is the vertical synchronization, the VSYNCS bit is set (active high). If the current display phase is the horizontal synchronization, the HSYNCS bit is active high.

The returned status does not depend on the configured polarity in the LTDC_GCR register, instead it returns the current active display phase.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **HSYNCS**: Horizontal synchronization display status
0: Active low
1: Active high

Bit 2 **VSYNCS**: Vertical synchronization display status
0: Active low
1: Active high

Bit 1 **HDES**: Horizontal data enable display status
0: Active low
1: Active high

Bit 0 **VDES**: Vertical data enable display status
0: Active low
1: Active high

34.7.18 LTDC layer x control register (LTDC_LxCR)

Address offset: 0x084 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUTEN	Res.	Res.	COLKEN	LEN
											rw			rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CLUTEN**: Color look-up table enable
This bit is set and cleared by software.
0: Color look-up table disable
1: Color look-up table enable
The CLUT is only meaningful for L8, AL44 and AL88 pixel format. Refer to *Color look-up table (CLUT)*

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **COLKEN**: Color keying enable
This bit is set and cleared by software.
0: Color keying disable
1: Color keying enable

Bit 0 **LEN**: Layer enable
This bit is set and cleared by software.
0: Layer disabled
1: Layer enabled

34.7.19 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR)

Address offset: 0x088 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the horizontal position (first and last pixel) of the layer 1 or 2 window.

The first visible pixel of a line is the programmed value of AHBP[11:0] bits + 1 in the LTDC_BPCR register.

The last visible pixel of a line is the programmed value of AAW[11:0] bits in the LTDC_AWCR register.

Example: The LTDC_BPCR register is configured to 0x000E0005 (AHBP[11:0] is 0xE) and the LTDC_AWCR register is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:

- layer window first pixel, WHSTPOS[11:0], must be programmed to 0x14 (0xE+1+0x5).
- layer window last pixel, WHSPPOS[11:0], must be programmed to 0x28A.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WHSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WHSPPOS[11:0]**: Window horizontal stop position
These bits configure the last visible pixel of a line of the layer window.
WHSPPOS[11:0] must be ≥ AHBP[11:0] bits + 1 (programmed in LTDC_BPCR register).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WHSTPOS[11:0]**: Window horizontal start position
These bits configure the first visible pixel of a line of the layer window.
WHSTPOS[11:0] must be ≤ AAW[11:0] bits (programmed in LTDC_AWCR register).

34.7.20 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR)

Address offset: 0x08C + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the vertical position (first and last line) of the layer1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[11:0] bits + 1 in the register LTDC_BPCR register.

The last visible line of a frame is the programmed value of AAH[11:0] bits in the LTDC_AWCR register.

Example:

The LTDC_BPCR register is configured to 0x000E0005 (AVBP[11:0] is 0x5) and the LTDC_AWCR register is configured to 0x028E01E5 (AAH[11:0] is 0x1E5).

To configure the vertical position of a window size of 630x460, with vertical start offset of eight lines in the active data area:

- layer window first line, WVSTPOS[11:0], must be programmed to 0xE (0x5 + 1 + 0x8).
- layer window last line, WVSPPOS[11:0], must be programmed to 0x1DA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WVSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WVSPPOS[11:0]**: Window vertical stop position
These bits configure the last visible line of the layer window.
WVSPPOS[11:0] must be ≥ AVBP[11:0] bits + 1 (programmed in LTDC_BPCR register).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WVSTPOS[11:0]**: Window vertical start position
These bits configure the first visible line of the layer window.
WVSTPOS[11:0] must be ≤ AAH[11:0] bits (programmed in LTDC_AWCR register).

34.7.21 LTDC layer x color keying configuration register (LTDC_LxCKCR)

Address offset: 0x090 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color key value (RGB), that is used by the color keying.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CKGREEN[7:0]								CKBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CKRED[7:0]**: Color key red value

Bits 15:8 **CKGREEN[7:0]**: Color key green value

Bits 7:0 **CKBLUE[7:0]**: Color key blue value

34.7.22 LTDC layer x pixel format configuration register (LTDC_LxPFCR)

Address offset: 0x094 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PF[2:0]**: Pixel format

These bits configure the pixel format
000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
101: L8 (8-bit luminance)
110: AL44 (4-bit alpha, 4-bit luminance)
111: AL88 (8-bit alpha, 8-bit luminance)

34.7.23 LTDC layer x constant alpha configuration register (LTDC_LxCACR)

Address offset: 0x098 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 00FF

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending. Refer to LTDC_LxBFCR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CONSTA[7:0]**: Constant alpha

These bits configure the constant alpha used for blending. The constant alpha is divided by 255 by hardware.

Example: if the programmed constant alpha is 0xFF, the constant alpha value is 255 / 255 = 1.

34.7.24 LTDC layer x default color configuration register (LTDC_LxDCCR)

Address offset: 0x09C + 0x80 * (x - 1), (*x* = 1 to 2)

Reset value: 0x0000 0000

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCALPHA[7:0]								DCRED[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DCGREEN[7:0]								DCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DCALPHA[7:0]**: Default color alpha

These bits configure the default alpha value.

Bits 23:16 **DCRED[7:0]**: Default color red

These bits configure the default red value.

Bits 15:8 **DCGREEN[7:0]**: Default color green

These bits configure the default green value.

Bits 7:0 **DCBLUE[7:0]**: Default color blue

These bits configure the default blue value.

34.7.25 LTDC layer x blending factors configuration register (LTDC_LxBFCR)

Address offset: 0x0A0 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0607

This register defines the blending factors F1 and F2.

The general blending formula is: BC = BF1 x C + BF2 x Cs

- BC = blended color
- BF1 = blend factor 1
- C = current layer color
- BF2 = blend factor 2
- Cs = subjacent layers blended color

The constant alpha value, is the programmed value in LTDC_LxCACR divided by 255 by hardware.

Example: Only layer1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in LTDC_LxCACR is 240 (0xF0). Thus, the constant alpha value is 240 / 255 = 0.94. C: current layer color is 128.
Cs: background color is 48. Layer1 is blended with the background color.
BC = constant alpha x C + *(1 - Constant Alpha*) x Cs = *0.94 x 128 + (1- 0.94) x 48 = 123.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
					rw	rw	rw						rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **BF1[2:0]**: Blending factor 1
These bits select the blending factor F1.
100: constant alpha
110: pixel alpha x constant alpha
Others: Reserved

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **BF2[2:0]**: blending factor 2
These bits select the blending factor F2
101: 1 - constant alpha
111: 1 - (pixel alpha x constant alpha)
Others: Reserved

34.7.26 LTDC layer x color frame buffer address register (LTDC_LxCFBAR)

Address offset: 0x0AC + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color frame buffer start address which has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFBADD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFBADD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFBADD[31:0]**: Color frame buffer start address
These bits define the color frame buffer start address.

34.7.27 LTDC layer x color frame buffer length register (LTDC_LxCFBLR)

Address offset: 0x0B0 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color frame buffer line length and pitch.

Example:

- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 3 = 960), where pitch = line length requires a value of 0x03C003C7 to be written into this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CFBP[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	CFBLL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **CFBP[13:0]**: Color frame buffer pitch in bytes
These bits define the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CFBLL[13:0]**: Color frame buffer line length

These bits define the length of one line of pixels in bytes + 7.

The line length is computed as follows:

active high width * number of bytes per pixel + 7.

34.7.28 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR)

Address offset: 0x0B4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the number of lines in the color frame buffer.

The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt is generated if enabled.

The start address and pitch settings on the other hand define the correct start of every line in memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CFBLNBR[11:0]**: Frame buffer line number

These bits define the number of lines in the frame buffer that corresponds to the active high width.

34.7.29 LTDC layer x CLUT write register (LTDC_LxCLUTWR)

Address offset: 0x0C4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the CLUT address and the RGB value.

The CLUT write register must be configured only during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in the LTDC_LxCR register.

The CLUT is only meaningful for L8, AL44 and AL88 pixel format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLUTADD[7:0]								RED[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GREEN[7:0]								BLUE[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **CLUTADD[7:0]**: CLUT address
These bits configure the CLUT address (color position within the CLUT) of each RGB value.

Bits 23:16 **RED[7:0]**: Red value
These bits configure the red value.

Bits 15:8 **GREEN[7:0]**: Green value
These bits configure the green value.

Bits 7:0 **BLUE[7:0]**: Blue value
These bits configure the blue value.

34.7.30 LTDC register map

Table 247. LTDC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LTDC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER[7:0]								MINVER[7:0]								REV[7:0]							
	Reset value									0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0x004	LTDC_LCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNBR[7:0]							
	Reset value																									0	0	0	0	0	0	1	0
0x008	LTDC_SSCR	Res.	Res.	Res.	Res.	HSW[11:0]												Res.	Res.	Res.	Res.	VSH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x00C	LTDC_BPCR	Res.	Res.	Res.	Res.	AHBP[11:0]												Res.	Res.	Res.	Res.	AVBP[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x010	LTDC_AWCR	Res.	Res.	Res.	Res.	AAW[11:0]												Res.	Res.	Res.	Res.	AAH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x014	LTDC_TWCR	Res.	Res.	Res.	Res.	TOTALW[11:0]												Res.	Res.	Res.	Res.	TOTALH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN	Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	Res.	LTDCEN
	Reset value	0	0	0	0												0		0	1	0		0	1	0		0	1	0				0
0x01C	LTDC_GC1R	BMEN	Res.	STREN	DWP	SPP	IPP	TP	LNIP	BBEN	BCP	SHREN	Res.	GTC[2:0]			Res.	DT[1:0]		Res.	PRBEN	WRCH[3:0]				WGCH[3:0]				WBCH[3:0]			
	Reset value	0		1	0	1	0	1	1	1	1	1		0	0	1		1	1		1	1	0	0	0	1	0	0	0	1	0	0	0
0x020	LTDC_GC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDCA	BW[2:0]			DPAEN	DVAEN	STSAEN	EDCEN
	Reset value																									0	0	1	1	0	0	0	0
0x024	LTDC_SRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
	Reset value																															0	0

Table 247. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	LTDC_BCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]								BCGREEN[7:0]								BCBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Reserved																															
0x034	LTDC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIE	TERRIE	FUIE	LIE
	Reset value																													0	0	0	0
0x038	LTDC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRIF	TERRIF	FUIF	LIF
	Reset value																													0	0	0	0
0x03C	LTDC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRRIF	CTERRIF	CFUIF	CLIF
	Reset value																													0	0	0	0
0x040	LTDC_LIPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x044	LTDC_CPSR	CXPOS[15:0]																CYPOS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	LTDC_CDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
	Reset value																													1	1	1	1
0x04C-0x080	Reserved	Reserved																															
0x084	LTDC_L1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUTEN	Res.	Res.	COLKEN	LEN
	Reset value																												0			0	0
0x088	LTDC_L1WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x08C	LTDC_L1WVPCR	Res.	Res.	Res.	Res.	WVSPPOS[11:0]												Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x090	LTDC_L1CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	LTDC_L1PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x098	LTDC_L1CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x09C	LTDC_L1DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 247. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A0	LTDC_L1BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value																						1	1	0						1	1	1
0x0A4-0x0A8	Reserved	Reserved																															
0x0AC	LTDC_L1CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0	LTDC_L1CFBLR	Res.	Res.	CFBP[13:0]														Res.	Res.	CFBLL[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	LTDC_L1CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0B8-0x0C0	Reserved	Reserved																															
0x0C4	LTDC_L1CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8-0x100	Reserved	Reserved																															
0x104	LTDC_L2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUTEN	Res.	Res.	COLKEN	LEN
	Reset value																												0			0	0
0x108	LTDC_L2WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x10C	LTDC_L2WVPCR	Res.	Res.	Res.	Res.	WVSPPOS[11:0]												Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x110	LTDC_L2CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	LTDC_L2PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x118	LTDC_L2CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x11C	LTDC_L2DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x120	LTDC_L2BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value																						1	1	0						1	1	1
0x124-0x128	Reserved	Reserved																															
0x12C	LTDC_L2CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 247. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x130	LTDC_L2CFBLR	Res.	Res.	CFBP[13:0]														Res.	Res.	CFBLL[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134	LTDC_L2CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x138-0x140	Reserved	Reserved																															
0x144	LTDC_L2CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5* for the register boundary addresses.

35 True random number generator (RNG)

35.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG can be used to construct a NIST compliant deterministic random bit generator (DRBG), acting as a live entropy source.

The RNG true random number generator has been tested using the German BSI statistical tests of AIS-31 (T0 to T8).

35.2 RNG main features

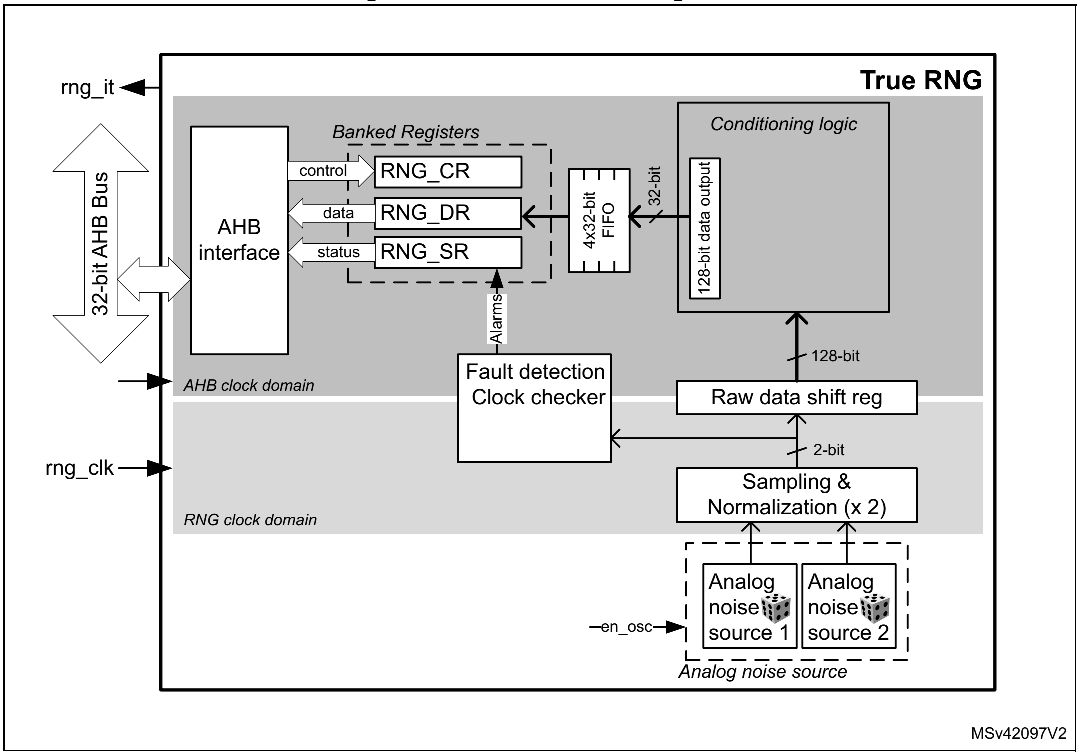
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source processed by a high quality conditioning stage.
- In the NIST configuration, it produces four 32-bit random samples every 16x $\frac{f_{AHB}}{f_{RNG}}$ AHB clock cycles, if value is higher than 213 cycles (213 cycles otherwise).
- It allows embedded continuous basic health tests with associated error management
  - Includes too low sampling clock detection and repetition count tests.
- It can be disabled to reduce power consumption.
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

35.3 RNG functional description

35.3.1 RNG block diagram

Figure 282 shows the RNG block diagram.

Figure 282. RNG block diagram



35.3.2 RNG internal signals

Table 248 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 248. RNG internal input/output signals

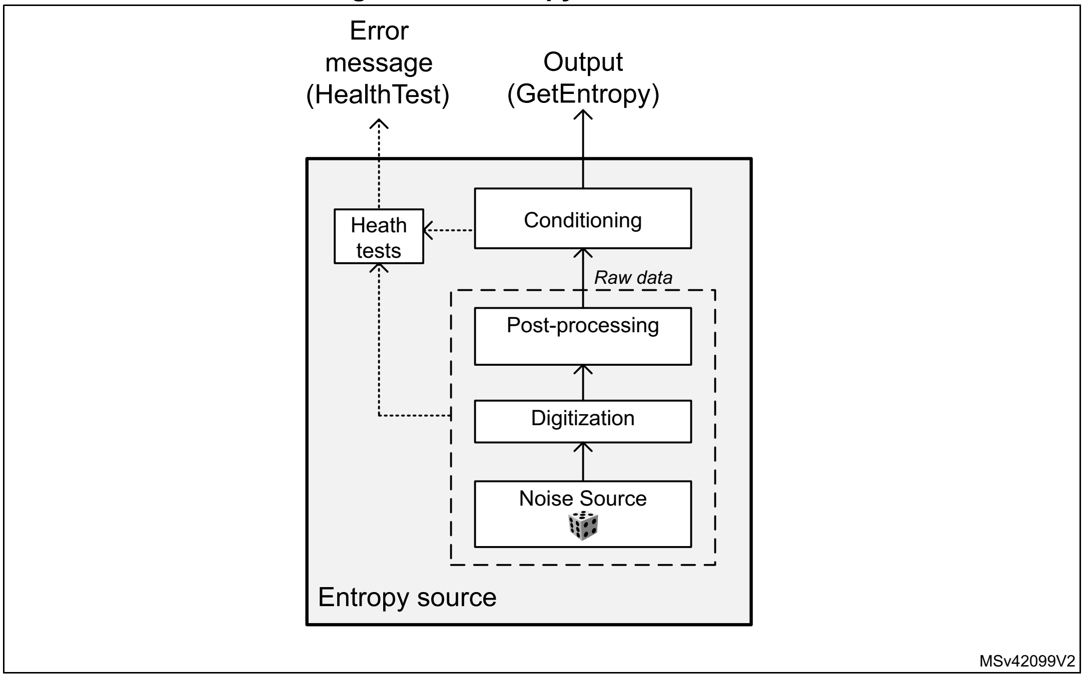
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk

35.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals. Within its boundary the RNG implements the entropy source model pictured on *Figure 283*.

It includes an analog noise source, a digitization stage with post-processing, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 283. Entropy source model



The components pictured above are detailed hereafter.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. It is composed of:

- Two analog noise sources, each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 35.3.8*.
- A sampling stage of these outputs clocked by a dedicated clock input (rng_clk), delivering a 2-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk).

Note: In Section 35.6 the recommended RNG clock frequencies are given.

Post processing

The sample values obtained from a true random noise source consist of 2-bit bitstrings. Because this noise source output is biased, the RNG implements a post-processing component that reduces that bias to a tolerable level.

More specifically, for each of the two noise source bits the RNG takes half of the bits from the sampled noise source, and half of the bits from the inverted sampled noise source. Thus, if the source generates more ‘1’ than ‘0’ (or the opposite), it is filtered.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit).

Also note that post-processing computations are triggered when at least 32 bits of raw datum is received and when output FIFO needs a refill. Thus, the RNG output entropy is maximum when the RNG 128-bit FIFO is emptied by application after 64 RNG clock cycles.

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in Section 35.5.

The conditioning component is clocked by the faster AHB clock.

Output buffer

A data output buffer can store up to four 32-bit words, which have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register 213 AHB clock cycles later.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features.

1. Continuous health tests, running indefinitely on the output of the noise source
   – Repetition count test, flagging an error when:
   a) One of the noise source has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bits patterns (“01” or “10”)
   b) Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bits patterns (“01” or “10”)
2. Vendor specific continuous test
   – Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle is smaller than AHB clock cycle divided by 32.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in Section 35.3.7.

Note: An interrupt can be generated when an error is detected.

35.3.4 RNG initialization

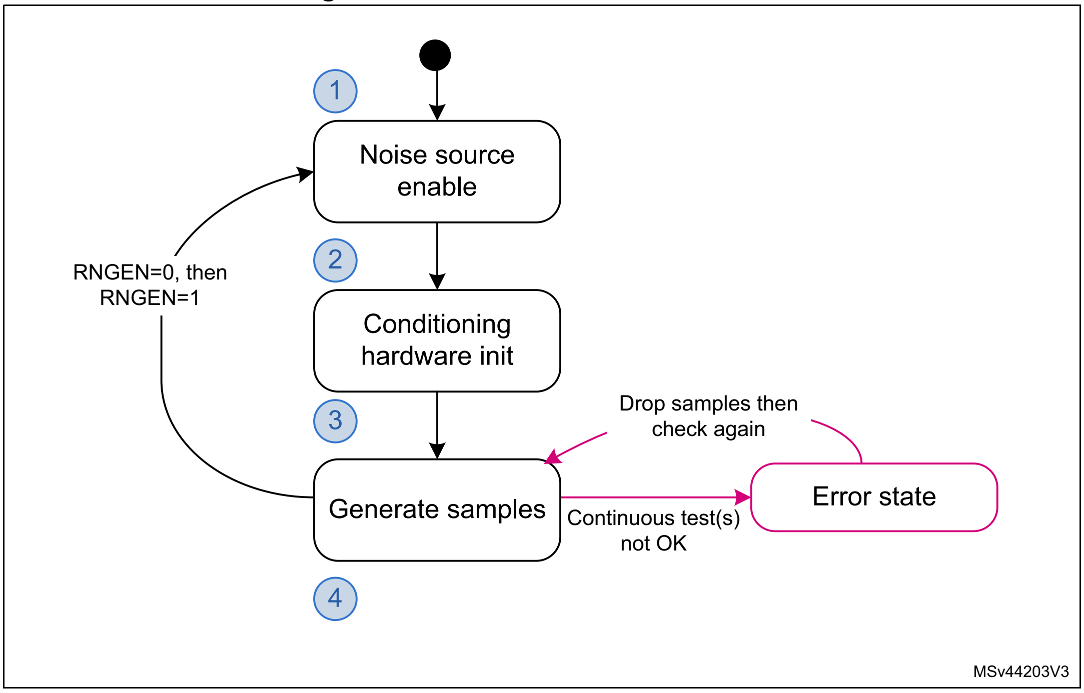
The RNG simplified state machine is pictured on *Figure 284*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and logic immediately starts sampling the analog output, filling the 128-bit conditioning shift register.
2. The conditioning logic is enabled and the post-processing context is initialized using two 128 noise source bits.
3. The conditioning stage internal input data buffer is filled again with 128-bit and one conditioning round is performed. The output buffer is then filled with the post processing result.
4. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 35.5*.

Figure 284. RNG initialization overview



35.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Enable the interrupts by setting the IE bit in the RNG_CR register. At the same time, enable the RNG by setting the bit RNGEN=1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 35.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Enable the random number generation by setting the RNGEN bit to “1” in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 35.3.7* must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

Low-power operation

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 35.3.8*.

Software post-processing

If a NIST approved DRBG with 128 bits of security strength is required an approved random generator software must be built around the RNG true random number generator.

Built-in health check functions are described in *Section 35.3.3: Random number generation*.

35.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock is used for noise source sampling. Recommended clock configurations are detailed in *Section 35.6: RNG entropy source validation*.

Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency the ***must be higher*** *than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 35.3.1: RNG block diagram* for details (AHB and RNG clock domains).

35.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 35.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

Note: The clock error has no impact on generated random numbers that is the application can still read the RNG_DR register.

CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy. If the error was detected during the initialization phase the whole initialization sequence is automatically restarted by the RNG.

The following sequence must be used to fully recover from a seed error after the RNG initialization:

1. Clear the SEIS bit by writing it to “0”.
2. Read out 12 words from the RNG_DR register, and discard each of them in order to clean the pipeline.
3. Confirm that SEIS is still cleared. Random number generation is back to normal.

35.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case the RNG must be re-enabled by the application until at least 32 new bits are collected from the noise source and a complete conditioning round is done. It corresponds to 16 RNG clock cycles to sample new bits, and 216 AHB clock cycles to run a conditioning round.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1.

35.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 35.3.7: Error management*
- Clock error, see *Section 35.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 249*.

Table 249. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

35.5 RNG processing time

The conditioning stage can produce four 32-bit random numbers every 16x $\frac{f_{AHB}}{f_{RNG}}$ clock cycles, if the value is higher than 213 cycles (213 cycles otherwise).
More time is needed for the first set of random numbers after the device exits reset (see

Section 35.3.4: RNG initialization). Indeed, after enabling the RNG for the first time, random data is first available after either:

- 128 RNG clock cycles + 426 AHB cycles, if $f_{AHB} < f_{threshold}$
- 192 RNG clock cycles + 213 AHB cycles, if $f_{AHB} \geq f_{threshold}$

With $f_{threshold} = (213 \times f_{RNG})/ 64$

35.6 RNG entropy source validation

35.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8). The results can be provided on demand or the customer can reproduce the tests.

35.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz (CED bit = ‘0’ in RNG_CR register) and rng_clk = 400 kHz (CED bit = ‘1’ in RNG_CR register).

Table 250. RNG configurations

<table>
<tr><th rowspan="2">Configuration</th><th colspan="6">RNG_CR bits</th><th rowspan="2">Loop number (N)</th><th rowspan="2">RNG_ HTCR register</th><th rowspan="2">RNG_ NSCR register</th></tr>
<tr><th>NISTC bit</th><th>RNG_ CONFIG1 [5:0]</th><th>CLKDIV [3:0]</th><th>RNG_ CONFIG2 [2:0](1)</th><th>RNG_ CONFIG3 [3:0](2)</th><th>CED bit</th></tr>
<tr><td>A</td><td colspan="9">Refer to NIST compliant RNG configuration table in AN4230 available from www.st.com.
This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on https://csrc.nist.gov/projects/cryptographic-module-validation-program.</td></tr>
<tr><td>B</td><td>1</td><td>0x18</td><td rowspan="2">0x0(3)</td><td>0x0</td><td>0x0</td><td>0</td><td>1</td><td rowspan="2">0x0000 NA(4)</td><td>default</td></tr>
<tr><td>C</td><td>0</td><td>0x0F</td><td>0x0</td><td>0xD</td><td>0</td><td>2</td><td>default</td></tr>
</table>

1. 0x1 value is recommended instead of 0x0 for RNG_CONFIG2[2:0], when RNG power consumption is critical. See the end of *Section 35.3.8* for details.
2. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4
3. The noise source sampling must be NA or less. Hence, if the RNG clock is different from NA, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 35.7.1* for details.
4. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).

35.6.3 Data collection

In order to run statistical tests, it is required to collect samples from the entropy source at the raw data level as well as at the output of the entropy source.For details on data collection and the running of statistical test suites refer to AN4230 “*STM32 microcontrollers random number generation validation using NIST statistical test suite*” available from *www.st.com*.

In bypass mode the bits [31:30] of the fourth word are always stuck at 0. Hence the continuous capture of samples is started from the fifth word.

35.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

35.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
										rw		rw	rw		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection
- 0: Clock error detection enabled
- 1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable
- 0: RNG interrupt is disabled
- 1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable
- 0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.
- 1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

35.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0. Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct (f_{RNGCLK}> f_{HCLK}/32)

1: The RNG is detected too slow (f_{RNGCLK}< f_{HCLK}/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequences has been detected:

– One of the noise sources has provided more than 64 consecutive bits at a constant value (“0” or “1”), or more than 32 consecutive occurrence of two bit patterns (“01” or “10”)

– Both noise sources have delivered more than 32 consecutive bits at a constant value (“0” or “1”), or more than 16 consecutive occurrence of two bit patterns (“01” or “10”)

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct (f_{RNGCLK}> f_{HCLK}/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow (f_{RNGCLK}< f_{HCLK}/32).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

35.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. After being read, this register delivers a new random value after 216 periods of AHB clock if the output FIFO is empty.

The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data
32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.
When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

35.7.4 RNG hardware configuration register (RNG_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:0 Reserved, must be kept at reset value.

35.7.5 RNG version register (RNG_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: RNG major revision

Bits 3:0 **MINREV[3:0]**: RNG minor revision

35.7.6 RNG identification register (RNG_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: RNG identification code

35.7.7 RNG size identification register (RNG_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: RNG size identification code
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1 Kbyte address decoding)

35.7.8 RNG register map

Table 251. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value																											0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC 0x20C	Reserved	Reserved																															
0x3F0	RNG_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.								Res.							
	Reset value																																

Table 251. RNG register map and reset map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F4	RNG_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	0	0	0	0	1
0x3F8	RNG_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x3FC	RNG_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

36 Cryptographic processor (CRYP)

36.1 Introduction

The cryptographic processor (CRYP) can be used both to encrypt and decrypt data using the DES, Triple-DES or AES algorithms. It is a fully compliant implementation of the following standards:

- The data encryption standard (DES) and Triple-DES (TDES) as defined by Federal Information Processing Standards Publication (FIPS PUB 46-3, Oct 1999), and the American National Standards Institute (ANSI X9.52)
- The advanced encryption standard (AES) as defined by Federal Information Processing Standards Publication (FIPS PUB 197, Nov 2001)

Multiple key sizes and chaining modes are supported:

- DES/TDES chaining modes ECB and CBC, supporting standard 56-bit keys with 8-bit parity per key
- AES chaining modes ECB, CBC, CTR, GCM, GMAC, CCM for key sizes of 128, 192 or 256 bits

The CRYP processor is a 32-bit AHB peripheral. It supports DMA transfers for incoming and outgoing data (two DMA channels are required). The peripheral also includes input and output FIFOs (each 8 words deep) for better performance.

36.2 CRYP main features

- Compliant implementation of the following standards:
  - NIST *FIPS publication 46-3, Data Encryption Standard (DES)*
  - ANSI *X9.52, Triple Data Encryption Algorithm Modes of Operation*
  - NIST *FIPS publication 197, Advanced Encryption Standard (AES)*
- AES symmetric block cipher implementation
  - 128-bit data block processing
  - Support for 128-, 192- and 256-bit cipher key lengths
  - Encryption and decryption with multiple chaining modes: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Counter mode (CTR), Galois Counter Mode (GCM), Galois Message Authentication Code mode (GMAC) and Counter with CBC-MAC (CCM)
  - 14 (respectively 18) clock cycles for processing one 128-bit block of data with a 128-bit (respectively 256-bit) key in AES-ECB mode
  - Integrated key scheduler with its key derivation stage (ECB or CBC decryption only)
- DES/TDES encryption/decryption implementation
  - 64-bit data block processing
  - Support for 64-, 128- and 192-bit cipher key lengths (including parity)
  - Encryption and decryption with support of ECB and CBC chaining modes
  - Direct implementation of simple DES algorithms (a single key K1 is used)

- 16 (respectively 48) clock cycles for processing one 64-bit block of data in DES (respectively TDES) ECB mode
- Software implementation of ciphertext stealing
- Features common to DES/TDES and AES
  - AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
  - 256-bit register for storing the cryptographic key (8x 32-bit registers)
  - 128-bit registers for storing initialization vectors (4× 32-bit)
  - 1x32-bit INPUT buffer associated with an internal IN FIFO of eight 32- bit words, corresponding to four incoming DES blocks or two AES blocks
  - 1x32-bit OUTPUT buffer associated with an internal OUT FIFO of eight 32-bit words, corresponding to four processed DES blocks or two AES blocks
  - Automatic data flow control supporting direct memory access (DMA) using two channels (one for incoming data, one for processed data). The OUT FIFO supports both single and burst transfers, while the IN FIFO supports only burst transfers.
  - Data swapping logic to support 1-, 8-, 16- or 32-bit data
  - Possibility for software to suspend a message if the cryptographic processor needs to process another message with higher priority (suspend/resume operation)

36.3 CRYP implementation

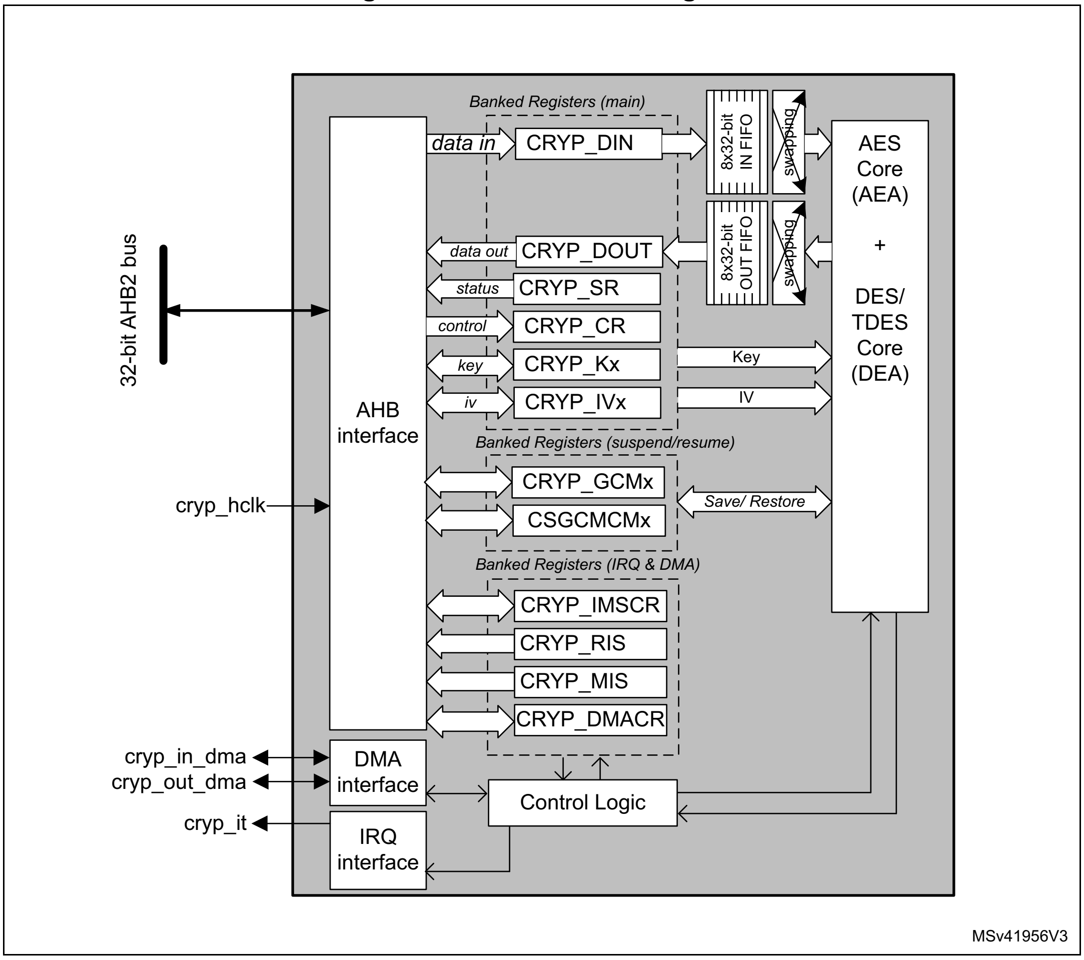
This device has one CRYP instance and to an MDMA peripheral.

36.4 CRYP functional description

36.4.1 CRYP block diagram

The figure below shows the block diagram of the cryptographic processor.

Figure 285. CRYP block diagram



36.4.2 CRYP internal signals

Table 252 provides a list of useful-to-know internal signals available at cryptographic processor level and not at STM32 product level (on pads).

Table 252. CRYP internal input/output signals

Signal name	Signal type	Description
cryp_hclk	Digital input	AHB bus clock
cryp_it	Digital output	Cryptographic processor global interrupt request
cryp_in_dma	Digital input/output	IN FIFO DMA burst request/ acknowledge
cryp_out_dma	Digital input/output	OUT FIFO DMA burst request/ acknowledge

36.4.3 CRYP DES/TDES cryptographic core

Overview

The DES/Triple-DES cryptographic core consists of three components:

- The DES Algorithm (DEA core)
- Multiple keys (one for the DES algorithm, one to three for the TDES algorithm)
- The initialization vector, which is used only in CBC mode

The DES/Triple-DES cryptographic core provides two operating modes:

- **ALGODIR = 0**: Plaintext encryption using the key stored in the CRYP_Kx registers.
- **ALGODIR = 1**: Ciphertext decryption using the key stored in the CRYP_Kx registers.

The operating mode is selected by programming the ALGODIR bit in the CRYP_CR register.

Typical data processing

Typical usage of the cryptographic processor in DES modes can be found in *Section 36.4.10: CRYP DES/TDES basic chaining modes (ECB, CBC)*.

Note: The outputs of the intermediate DEA stages are never revealed outside the cryptographic boundary, with the exclusion of the IV registers in CBC mode.

DES/TDES keying and chaining modes

The DES/TDES allows three different keying options:

- *Three independent keys*

  The first option specifies that all the keys are independent, that is, K1, K2 and K3 are independent. FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to this option as the Keying Option 1 and, to the TDES as 3-key TDES.
- *Two independent keys*

  The second option specifies that K1 and K2 are independent and K3 is equal to K1, that is, K1 and K2 are independent, K3 = K1. FIPS PUB 46-3 – 1999 (and ANSI X9.52

– 1998) refers to this second option as the Keying Option 2 and, to the TDES as 2-key TDES.

- *Three equal keys*

  The third option specifies that K1, K2 and K3 are equal, that is:

  K1 = K2 = K3

  FIPS PUB 46-3 – 1999 (and ANSI X9.52 – 1998) refers to the third option as the Keying Option 3. This "1-key" TDES is equivalent to single DES.

The following chaining algorithms are supported by the DES hardware and can be selected through the ALGOMODE bits in the CRYP_CR register:

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)

These modes are described in details in *Section 36.4.10: CRYP DES/TDES basic chaining modes (ECB, CBC)*.

36.4.4 CRYP AES cryptographic core

Overview

The AES cryptographic core consists of the following components:

- The AES algorithm (AEA core)
- The Multiplier over a binary Galois field (GF2mul)
- The key information
- The initialization vector (IV) or Nonce information
- Chaining algorithms logic (XOR, feedback/counter, mask)

The AES core works on 128-bit data blocks of (four words) with 128-, 192- or 256-bit key lengths. Depending on the chaining mode, the peripheral requires zero or one 128-bit initialization vector (IV).

The cryptographic peripheral features two operating modes:

- **ALGODIR = 0**: Plaintext encryption using the key stored in the CRYP_Kx registers.
- **ALGODIR = 1**: Ciphertext decryption using the key stored in the CRYP_Kx registers. When ECB and CBC chaining modes are selected, an initial key derivation process is automatically performed by the cryptographic peripheral.

The operating mode is selected by programming the ALGODIR bit in the CRYP_CR register.

Typical data processing

A description of cryptographic processor typical usage in AES mode can be found in *Section 36.4.11: CRYP AES basic chaining modes (ECB, CBC)*.

Note: The outputs of the intermediate AEA stages is never revealed outside the cryptographic boundary, with the exclusion of the IV registers.

AES chaining modes

The following chaining algorithms are supported by the cryptographic processor and can be selected through the ALGOMODE bits in the CRYP_CR register:

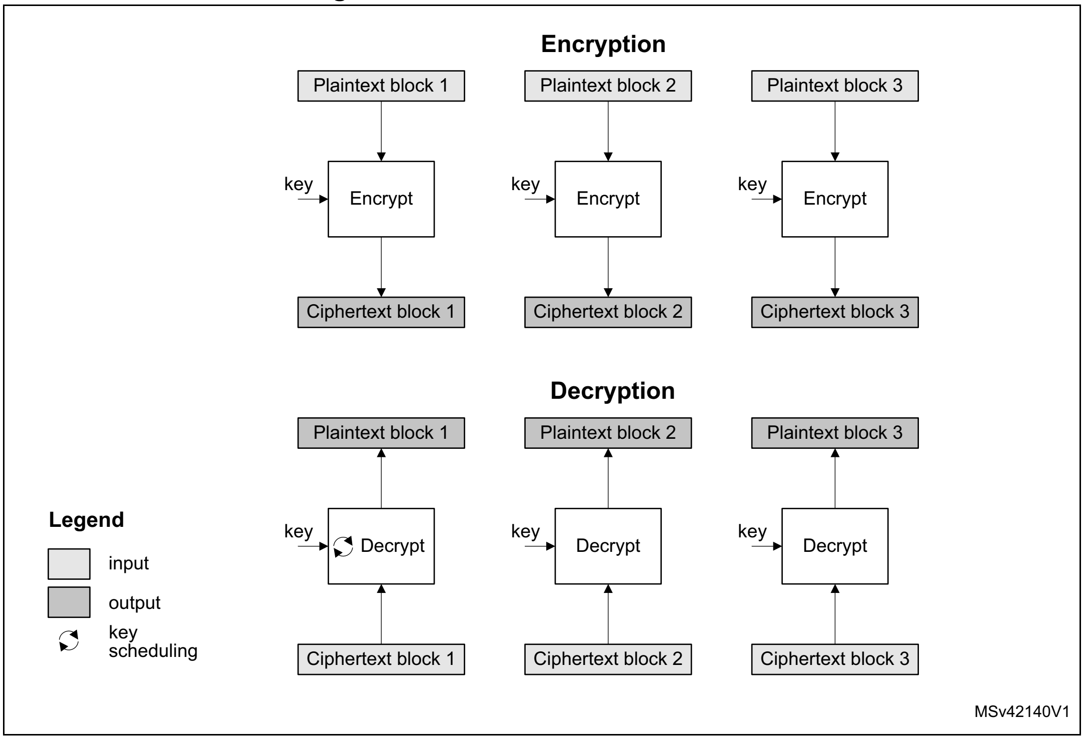
- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter Mode (CTR)
- Galois/Counter Mode (GCM)
- Galois Message Authentication Code mode (GMAC)
- Counter with CBC-MAC (CCM)

A quick introduction on these chaining modes can be found in the following subsections.

For detailed instructions, refer to Section 36.4.11: CRYP AES basic chaining modes (ECB, CBC) and onward.

AES Electronic CodeBook (ECB)

Figure 286. AES-ECB mode overview

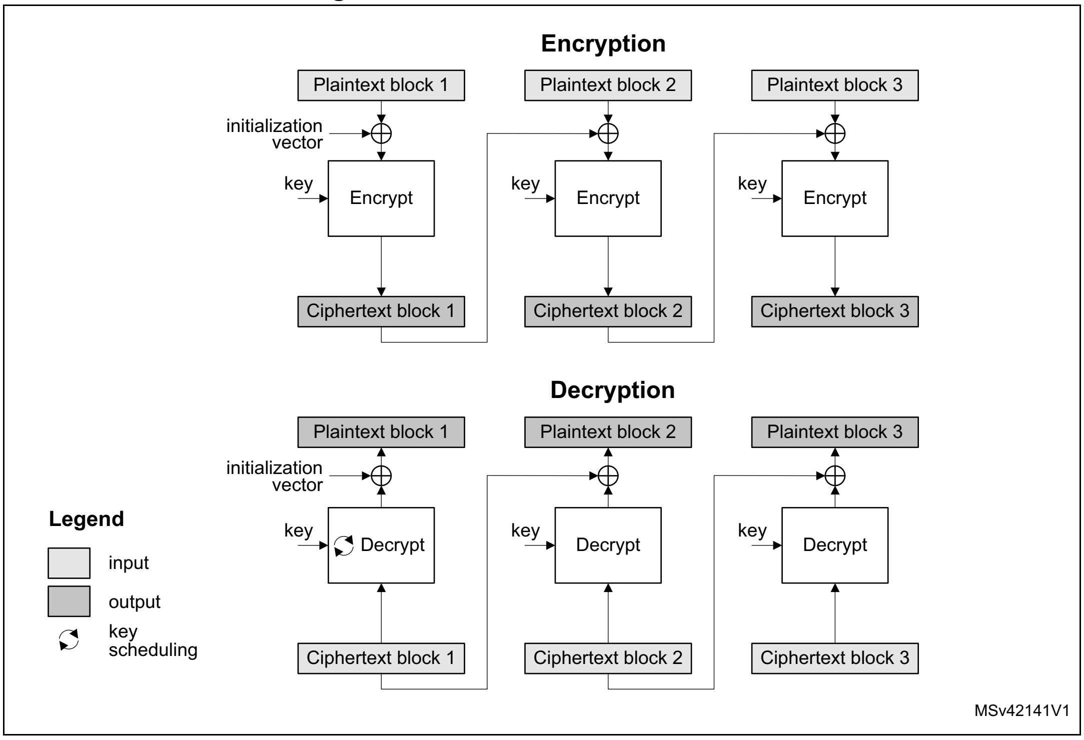


ECB is the simplest operating mode. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately.

Note: For decryption, a special key scheduling is required before processing the first block.

AES Cipher block chaining (CBC)

Figure 287. AES-CBC mode overview

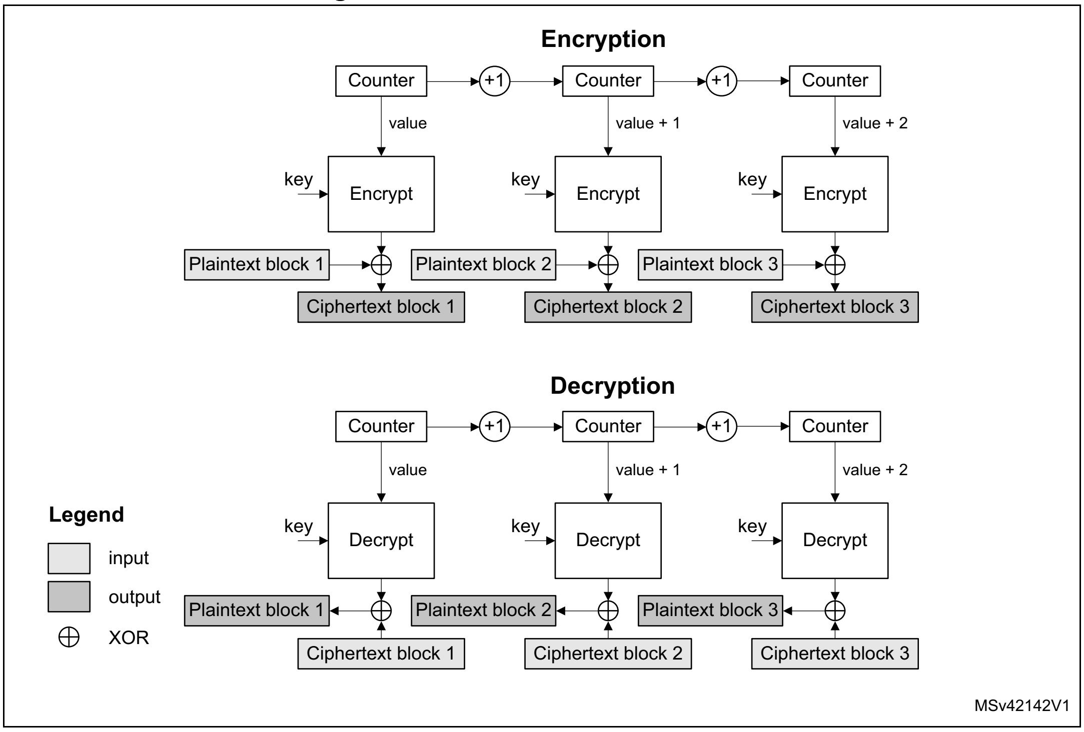


CBC operating mode chains the output of each block with the input of the following block. To make each message unique, an initialization vector is used during the first block processing.

Note: For decryption, a special key scheduling is required before processing the first block.

AES Counter mode (CTR)

Figure 288. AES-CTR mode overview

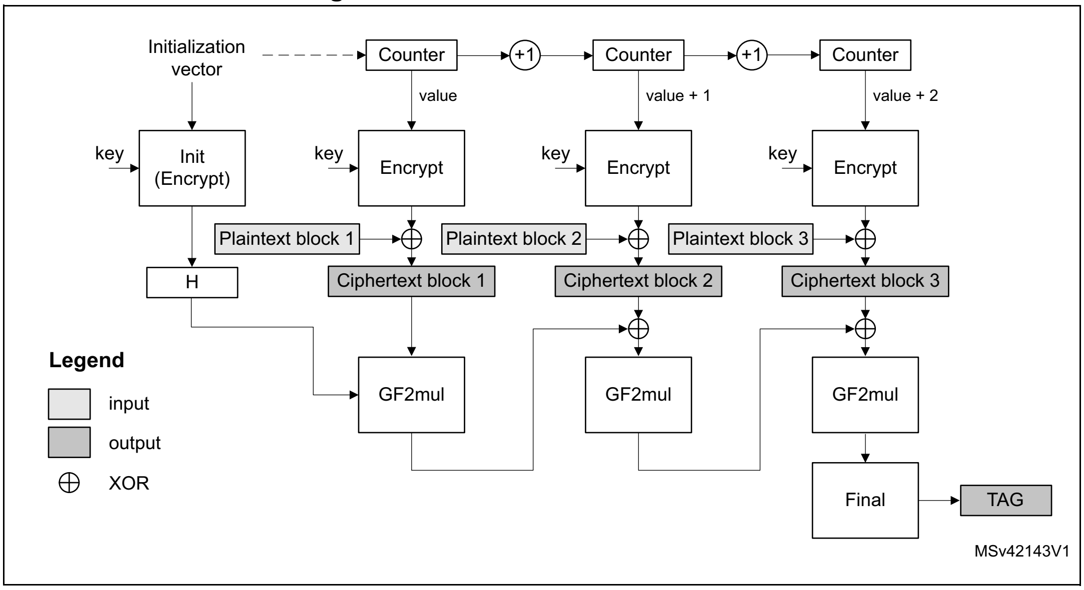


The CTR mode uses the AES core to generate a key stream; these keys are then XORed with the plaintext to obtain the ciphertext as specified in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Note: Unlike ECB and CBC modes, no key scheduling is required for the CTR decryption, since in this chaining scheme the AES core is always used in encryption mode for producing the counter blocks.

AES Galois/Counter mode (GCM)

Figure 289. AES-GCM mode overview

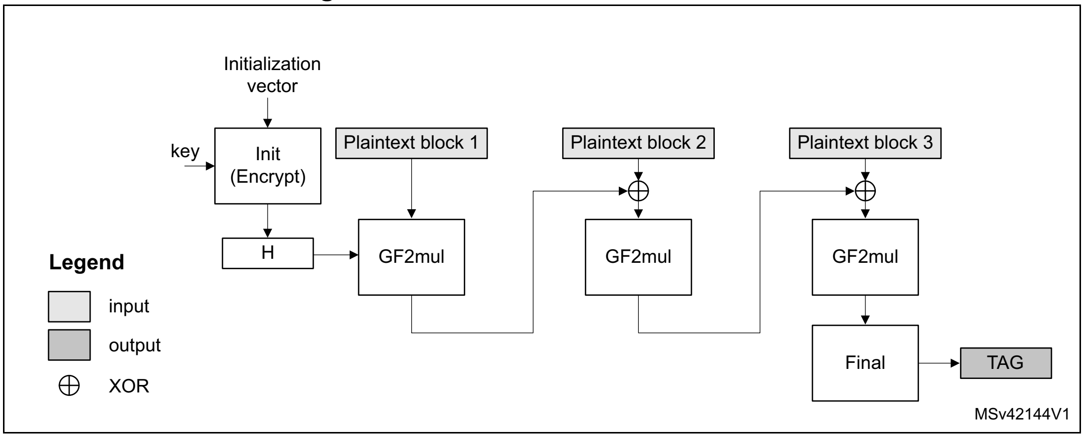


In Galois/Counter mode (GCM), the plaintext message is encrypted, while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and its MAC (also known as authentication tag). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. It requires an initial value and a particular 128-bit block at the end of the message.

AES Galois message authentication code (GMAC)

Figure 290. AES-GMAC mode overview

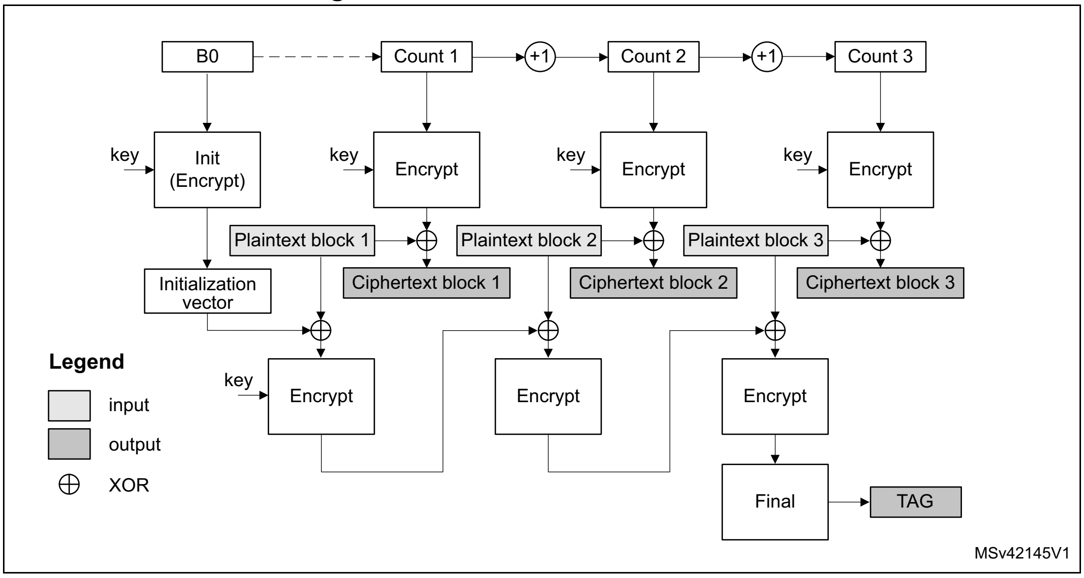


Galois Message Authentication Code (GMAC) allows authenticating a message and generating the corresponding message authentication code (MAC). It is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*.

GMAC is similar to Galois/Counter mode (GCM), except that it is applied on a message composed only by clear-text authenticated data (i.e. only header, no payload).

AES Counter with CBC-MAC (CCM)

Figure 291. AES-CCM mode overview



In Counter with Cipher Block Chaining-Message Authentication Code (CCM), the plaintext message is encrypted while a message authentication code (MAC) is computed in parallel, thus generating the corresponding ciphertext and the corresponding MAC (also known as tag). It is described by NIST in *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

CCM mode is based on AES in counter mode for confidentiality and it uses CBC for computing the message authentication code. It requires an initial value.

Like GCM CCM chaining mode, AES-CCM mode can be applied on a message composed only by cleartext authenticated data (i.e. only header, no payload). Note that this way of using CCM is not called CMAC (it is not similar to GCM/GMAC), and its usage is not recommended by NIST.

36.4.5 CRYP procedure to perform a cipher operation

Introduction

To understand how the cryptographic peripheral operates, a typical cipher operation is described below. For the detailed peripheral usage according to the cipher mode, refer to the specific section, for example *Section 36.4.11: CRYP AES basic chaining modes (ECB, CBC)*.

CRYP initialization

To initialize the cryptographic processor, first disable it by clearing the CRYPEN bit in the CRYP_CR register. Then execute the following steps:

1. Configure the algorithm and the chaining mode through the ALGOMODE and ALGODIR bits in the CRYP_CR register. Configure also the key size with the KEYSIZE bits.
   – When ALGODIR is set to 1 (decryption) and the selected algorithm/chaining mode is AES-ECB or AES-CBC, an initial key derivation of the encryption key must be performed as described in *Section 36.4.7: Preparing the CRYP AES key for decryption*.
2. When the previous step is complete, and when applicable, write the symmetric key into the CRYP_KxL/R registers. The way to write key registers is defined in *Section 36.4.17: CRYP key registers*
3. Configure the data type (1, 8, 16 or 32 bits) through the DATATYPE bits of the CRYP_CR register.
4. When it is required (for example for CBC or CTR chaining modes), write the initialization vectors into the CRYP_IVx(L/R)R register.
5. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register.

Preliminary warning for all cases

If the ECB or CBC mode is selected and data are not a multiple of 64 bits (for DES) or 128 bits (for AES), the second and the last block management is more complex than the sequences below. Refer to *Section 36.4.8: CRYP stealing and data padding* for more details.

Appending data using the CPU in Polling mode

1. Enable the cryptographic processor by setting to 1 the CRYPEN bit in the CRYP_CR register.
2. Write data in the IN FIFO (one block or until the FIFO is full).
3. Repeat the following sequence until the second last block of data has been processed:
   a) Wait until the not-empty-flag OFNE is set to 1, then read the OUT FIFO (one block or until the FIFO is empty).
   b) Wait until the not-full-flag IFNF is set to 1, then write the IN FIFO (one block or until the FIFO is full) except if it is the last block.
4. The BUSY bit is set automatically by the cryptographic processor. At the end of the processing, the BUSY bit returns to 0 and both FIFOs are empty (IN FIFO empty flag IFEM = 1 and OUT FIFO not empty flag OFNE = 0).
5. If the next processing block is the last block, the CPU must pad (when applicable) the data with zeroes to obtain a complete block and specify the number of non-valid bytes using NPBLB bits in CRYP_CR register in case of AES GCM payload encryption or AES CCM payload decryption (otherwise the tag computation will be wrong). This operation must be performed after checking that the BUSY bit in the CRYP_CR register is set to 0.

Note: NPBLB bits are not used in the header phase of AES GCM, GMAC and CCM modes.

6. When the operation is complete, the cryptographic processor can be disabled by clearing the CRYPEN bit in CRYP_CR register.

Appending data using the CPU in Interrupt mode

1. Enable the interrupts by setting the INIM and OUTIM bits in the CRYP_IMSCR register.
2. Enable the cryptographic processor by setting to 1 the CRYPEN bit in the CRYP_CR register.
3. In the interrupt service routine that manages the input data:
   a) If the last block is being loaded, the CPU must pad (when applicable) the data with zeroes to have a complete block and specify the number of non-valid bytes using NPBLB bits in CRYP_CR register in case of AES GCM payload encryption or AES CCM payload decryption (otherwise the tag computation will be wrong). This operation must be performed after checking that the BUSY bit in the CRYP_CR register is set to 0. Then load the block into the IN FIFO.

Note: NPBLB bits are not used in the header phase of AES GCM, GMAC and CCM.

   b) If it is not the last block, load the data into the IN FIFO. You can load only one block (2 words for DES, 4 words for AES), or load data until the FIFO is full.
   c) In all cases, after the last word of data has been written, disable the interrupt by clearing the INIM interrupt mask.
4. In the interrupt service routine that manages the input data:
   a) Read the output data from the OUT FIFO. You can read only one block (2 words for DES, 4 words for AES), or read data until the FIFO is empty.
   b) When the last word has been read, INIM and BUSY bits are set to 0 and both FIFOs are empty (IFEM = 1 and OFNE = 0). You can disable the interrupt by clearing the OUTIM bit, and disable the peripheral by clearing the CRYPEN bit.
   c) If you read the last block of cleartext data (i.e. decryption), optionally discard the data that is not part of message/payload.

Appending data using the DMA

1. Prepare the last block of data by optionally padding it with zeroes to have a complete block.
2. Configure the DMA controller to transfer the input data from the memory and transfer the output data from the peripheral to the memory, as described in *Section 36.4.19: CRYP DMA interface*. The DMA should be configured to set an interrupt on transfer completion to indicate that the processing is complete. In case of AES GCM payload encryption or AES CCM payload decryption, DMA transfers must not include the last block. The sequence using the CPU described above must be used instead for this last block, because NPBLB bits needs to be setup before processing the block (otherwise the tag computation will be wrong).

Note: NPBLB bits are not used in the header phase of AES GCM, GMAC and CCM.

3. Enable the cryptographic processor by setting to 1 the CRYPEN bit in CRYP_CR register, then enable the DMA IN and OUT requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.
4. All the transfers and processing are managed by the DMA and the cryptographic processor. The DMA interrupt indicates that the processing is complete. Both FIFOs are normally empty and BUSY flag is set 0.

Note: DMA cannot be used when DES/TDES algorithms are selected.

Caution: It is important that DMA controller empties the cryptographic processor output FIFO before filling up the cryptographic processor input FIFO. To achieve this, the DMA controller should be configured so that the transfer from the cryptographic peripheral to the memory has a higher priority than the transfer from the memory to the cryptographic peripheral.

36.4.6 CRYP busy state

The cryptographic processor is busy and processing data (BUSY set to 1 in CRYP_SR register) when all the conditions below are met:

- CRYPEN = 1 in CRYP_CR register.
- There are enough data in the input FIFO (at least two words for the DES or TDES algorithm mode, four words for the AES algorithm mode).
- There is enough free-space in the output FIFO (at least two word locations for DES, four for AES).

Write operations to the CRYP_Kx(L/R)R key registers, to the CRYP_IVx(L/R)R initialization registers, or to bits [9:2] of the CRYP_CR register, are ignored when cryptographic processor is busy (i.e. the registers are not modified). It is thus not possible to modify the configuration of the cryptographic processor while it is processing a data block.

It is possible to clear the CRYPEN bit while BUSY bit is set to 1. In this case the ongoing DES/TDES or AES processing first completes (i.e. the word results are written to the output FIFO) before the BUSY bit is cleared by hardware.

Note: If the application needs to suspend a message to process another one with a higher priority, refer to Section 36.4.9: CRYP suspend/resume operations

When a block is being processed in DES or TDES mode, if the output FIFO becomes full and the input FIFO contains at least one new block, then the new block is popped off the input FIFO and the BUSY bit remains high until there is enough space to store this new block into the output FIFO.

36.4.7 Preparing the CRYP AES key for decryption

When performing an AES **ECB or CBC decryption**, the AES key has to be prepared. Indeed, in AES encryption the round 0 key is the one stored in the key registers, and AES decryption must start using the last round key. Hence, as the encryption key is stored in memory, a special key scheduling must be performed to obtain the decryption key. This preparation is not required in any other AES modes than ECB or CBC decryption.

When the cryptographic processor is disabled (CRYPEN = 0), the CRYP key preparation process is performed as follows:

1. Program ALGOMODE bits to 0x7 and ALGODIR to 0x0 in CRYP_CR. In addition, configure the key length with the KEYSIZE bits.
2. Write the symmetric key to the CRYP_KxL/R registers, as described in *Section 36.4.17: CRYP key registers*.
3. Enable the cryptographic processor by setting the CRYPEN bit in the CRYP_CR register. It immediately starts an AES round for key preparation (BUSY = 1).
4. Wait until the BUSY bit is cleared in the CRYP_SR register. Then update ALGOMODE bits in the CRYP_CR register to select the correct chaining mode, that is 0x4 for ECB or 0x5 for CBC.
5. The AES key is available in the CRYP key registers, ready to use for decryption.

Note: As the CRYPEN bitfield is reset by hardware at the end of the key preparation, the application software must set it again for the next operation.

The latency of the key preparation operation is 14, 16 or 18 clock cycles depending on the key size (128, 192 or 256 bits).

36.4.8 CRYP stealing and data padding

When using DES or AES algorithm in **ECB** or **CBC** modes to manage messages that are not multiple of the block size (64 bits for DES, 128 bits for AES), use ciphertext stealing techniques such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since the cryptographic processor does not implement such techniques, **the last two blocks** must be handled in a special way by the application.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, when the AES algorithm is used in other modes than ECB or CBC, incomplete input data blocks (i.e. block shorter than 128 bits) have to be padded with zeroes by the application prior to encryption (i.e. extra bits should be appended to the trailing end of the data string). After decryption, the extra bits have to be discarded. The cryptographic processor does not implement automatic data padding operation to **the last block**, so the application should follow the recommendation given in *Section 36.4.5: CRYP procedure to perform a cipher operation* to manage messages that are not multiple of 128 bits.

Note: Padding data are swapped in a similar way as normal data, according to the DATATYPE field in CRYP_CR register (see Section 36.4.16: CRYP data registers and data swapping for details).

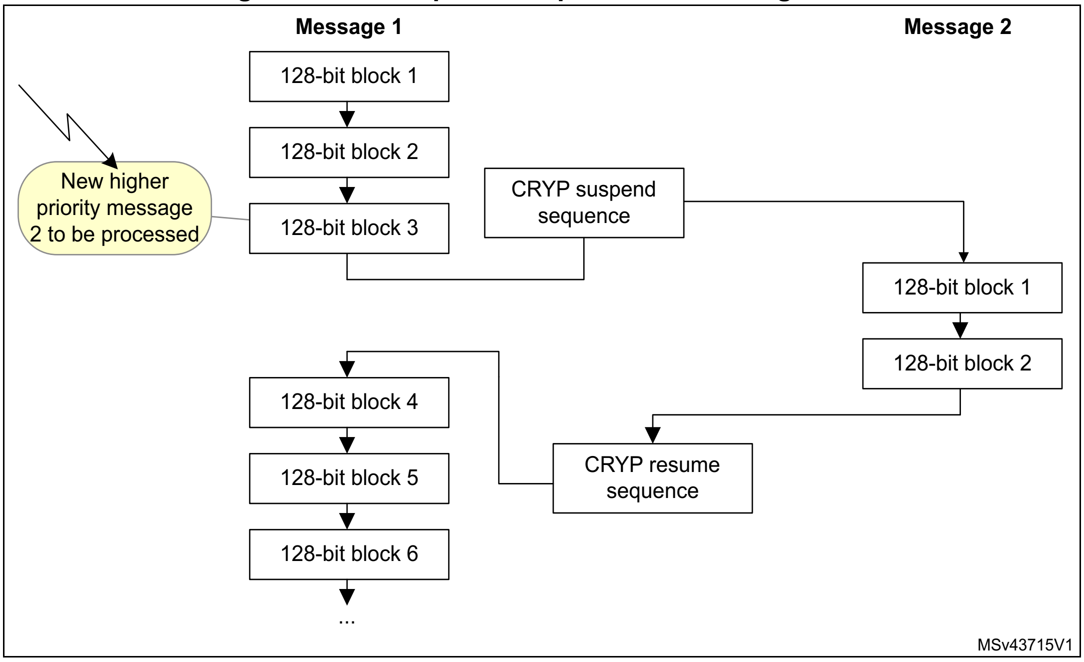
36.4.9 CRYP suspend/resume operations

A message can be suspended if another message with a higher priority has to be processed. When this highest priority message has been sent, the suspended message can be resumed in both encryption or decryption mode.

Suspend/resume operations do not break the chaining operation and the message processing can be resumed as soon as cryptographic processor is enabled again to receive the next data block.

Figure 292 gives an example of suspend.resume operation: message 1 is suspended in order to send a higher priority message (message 2), which is shorter than message 1 (AES algorithm).

Figure 292. Example of suspend mode management



A detailed description of suspend/resume operations can be found in each AES mode section.

36.4.10 CRYP DES/TDES basic chaining modes (ECB, CBC)

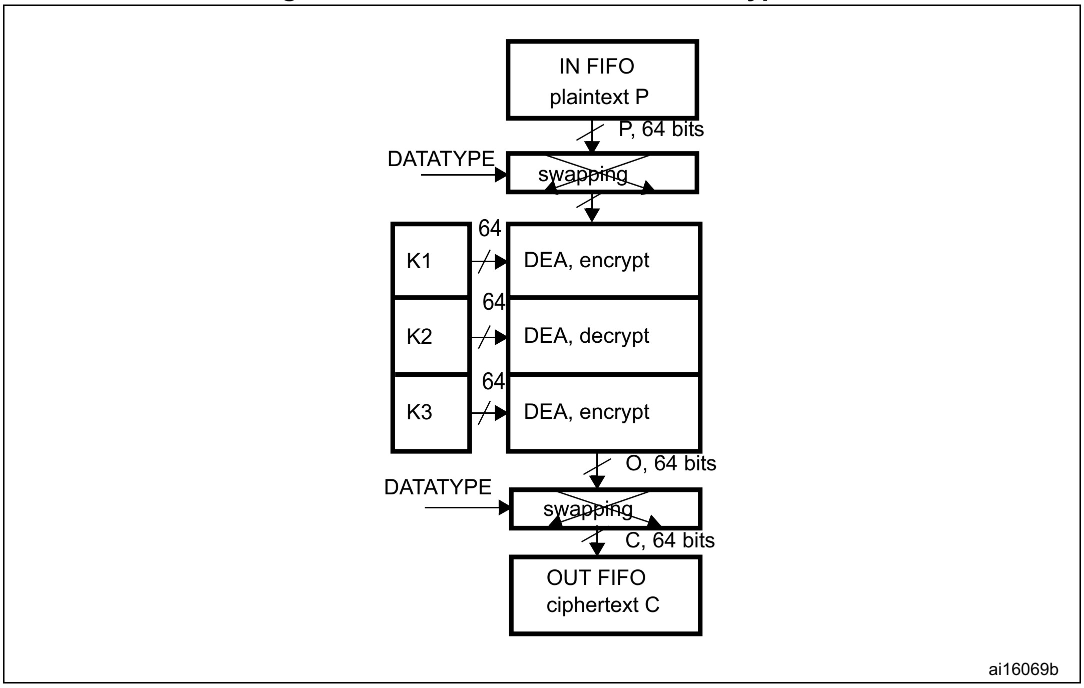
Overview

FIPS PUB 46-3 – 1999 (and ANSI X9.52-1998) provides a thorough explanation of the processing involved in the four operation modes supplied by the DES computing core: TDES-ECB encryption, TDES-ECB decryption, TDES-CBC encryption and TDES-CBC decryption. This section only gives a brief explanation of each mode.

DES/TDES-ECB encryption

Figure 293 illustrates the encryption in DES and TDES Electronic CodeBook (DES/TDES-ECB) mode. This mode is selected by programming ALGOMODE 0x0 and ALGODIR to 0 in CRYP_CR.

Figure 293. DES/TDES-ECB mode encryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

A 64-bit plaintext data block (P) is used after bit/byte/half-word as the input block (I). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA where the DES is performed in the decrypt state using K2. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K3. The resultant 64-bit output block (O) is used, after bit/byte/half-word swapping, as ciphertext (C) and it is pushed into the OUT FIFO.

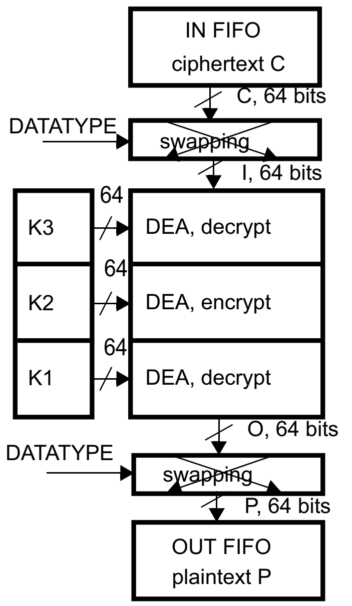
Note: For more information on data swapping, refer to Section 36.4.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 36.4.5: CRYP procedure to perform a cipher operation.

DES/TDES-ECB mode decryption

Figure 294 illustrates the decryption in DES and TDES Electronic CodeBook (DES/TDES-ECB) mode. This mode is selected by programming ALGOMODE to 0x0 and ALGODIR to 1 in CRYP_CR.

Figure 294. DES/TDES-ECB mode decryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

A 64-bit ciphertext block (C) is used, after bit/byte/half-word swapping, as the input block (I). The keying sequence is reversed compared to that used in the encryption process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed back directly to the input of the DEA where the DES is performed in the encrypt state using K2. The new result is directly fed to the input of the DEA where the DES is performed in the decrypt state using K1. The resultant 64-bit output block (O), after bit/byte/half-word swapping, produces the plaintext (P).

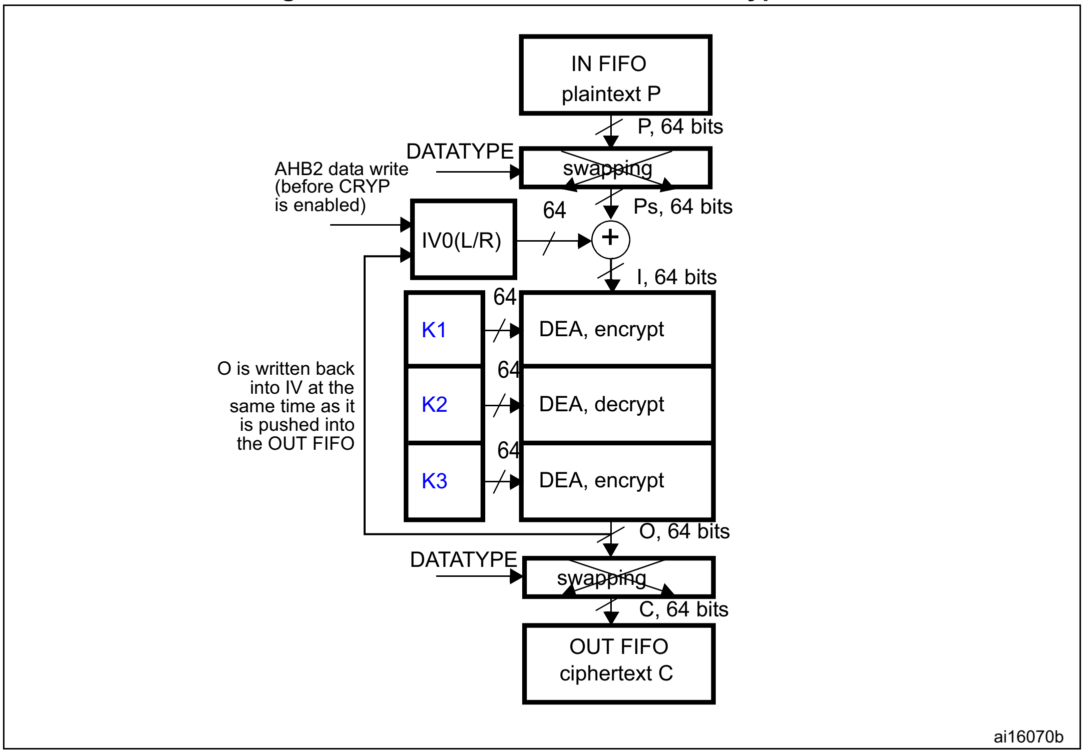
Note: For more information on data swapping refer to Section 36.4.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 36.4.5: CRYP procedure to perform a cipher operation.

DES/TDES-CBC encryption

Figure 295 illustrates the encryption in DES and TDES Cipher Block Chaining (DES/TDES-ECB) mode. This mode is selected by programming ALGOMODE to 0x1 and ALGODIR to 0 in CRYP_CR.

Figure 295. DES/TDES-CBC mode encryption



K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

This mode begins by dividing a plaintext message into 64-bit data blocks. In TCBC encryption, the first input block (I_1), obtained after bit/byte/half-word swapping, is formed by exclusive-ORing the first plaintext data block (P_1) with a 64-bit initialization vector IV ($I_1 = IV \oplus P_1$). The input block is processed through the DEA in the encrypt state using K1. The output of this process is fed back directly to the input of the DEA, which performs the DES in the decrypt state using K2. The output of this process is fed directly to the input of the DEA, which performs the DES in the encrypt state using K3. The resultant 64-bit output block (O_1) is used directly as the ciphertext (C_1), that is, $C_1 = O_1$.

This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the TDEA to produce the second ciphertext block.

This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted.

If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application.

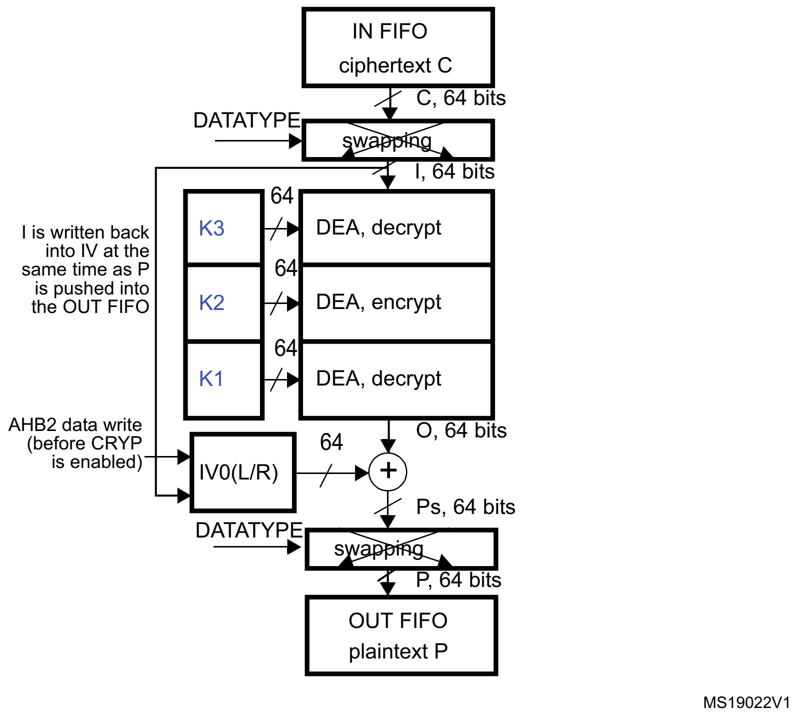
Note: For more information on data swapping refer to Section 36.4.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 36.4.5: CRYP procedure to perform a cipher operation.

DES/TDES-CBC decryption

Figure 295 illustrates the decryption in DES and TDES Cipher Block Chaining (DES/TDES-ECB) mode. This mode is selected by writing ALGOMODE to 0x1 and ALGODIR to 1 in CRYP_CR.

Figure 296. DES/TDES-CBC mode decryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: initialization vectors.

In this mode the first ciphertext block (C_1) is used directly as the input block (I_1). The keying sequence is reversed compared to that used for the encrypt process. The input block is processed through the DEA in the decrypt state using K3. The output of this process is fed directly to the input of the DEA where the DES is processed in the encrypt state using K2. This resulting value is directly fed to the input of the DEA where the DES is processed in the decrypt state using K1. The resulting output block is exclusive-ORed with the IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus IV$).

The second ciphertext block is then used as the next input block and is processed through the TDEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). Note that P_2 and O_2 refer to the second block of data.

The DES/TDES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

Ciphertext representing a partial data block must be decrypted in a manner specified for the application.

Note: For more information on data swapping refer to Section 36.4.16: CRYP data registers and data swapping.

Detailed DES/TDES encryption sequence can be found in Section 36.4.5: CRYP procedure to perform a cipher operation.

DES/TDES suspend/resume operations in ECB/CBC modes

Before interrupting the current message, the user application must respect the following steps:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM = 1 and OFNE = 0 in the CRYP_SR) and the BUSY bit is cleared. Alternatively, as the input FIFO can contain up to four unprocessed DES blocks, the application could decide for real-time reason to interrupt the cryptographic processing without waiting for the IN FIFO to be empty. In this case, the alternative is:
   a) Wait until OUT FIFO is empty (OFNE = 0).
   b) Read back the data loaded in the IN FIFO that have not been processed and save them in the memory until the IN FIFO is empty.
3. If DMA is used stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2] in the CRYP_CR register). If CBC mode is selected, save the initialization vector registers, since CRYP_IVx(L/R)R registers have changed from initial values during the data processing.

Note: Key registers do not need to be saved as the original key value is known by the application.

5. If DMA is used, save the DMA controller status (such as the pointers to IN and OUT data transfers, number of remaining bytes).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be 0).
3. Configure again the cryptographic processor with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. If the CBC mode is selected, restore CRYP_IVx(L/R)R registers using the saved configuration.
5. Optionally, write the data that were saved during context saving into the IN FIFO.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again DMA requests for the cryptographic processor, by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Caution: DMA cannot be used when DES/TDES algorithms are selected.

36.4.11 CRYP AES basic chaining modes (ECB, CBC)

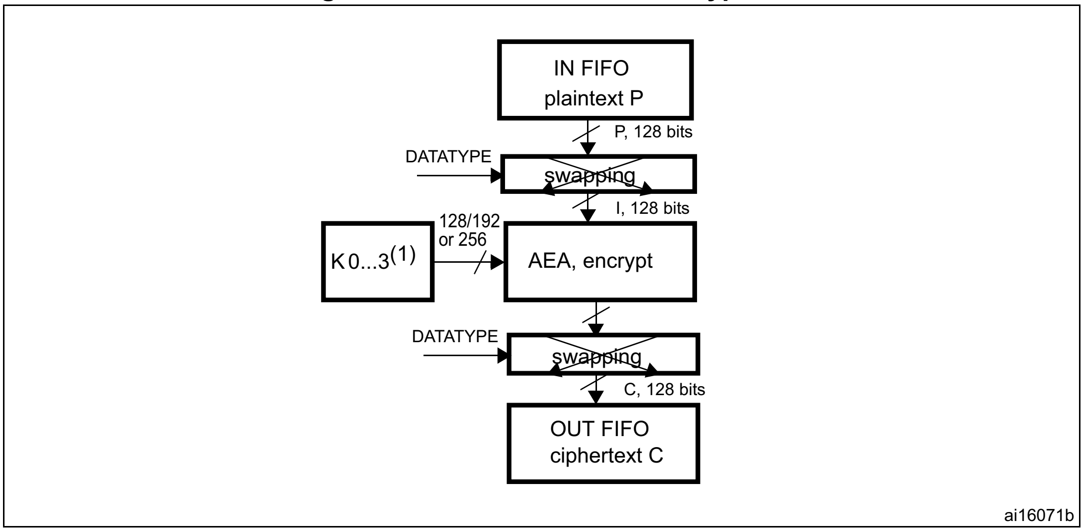
Overview

FIPS PUB 197 (November 26, 2001) provides a thorough explanation of the processing involved in the four basic operation modes supplied by the AES computing core: AES-ECB encryption, AES-ECB decryption, AES-CBC encryption and AES-CBC decryption. This section only gives a brief explanation of each mode.

AES ECB encryption

Figure 297 illustrates the AES Electronic codebook (AES-ECB) mode encryption. This mode is selected by writing ALGOMODE to 0x4 and ALGODIR to 0 in CRYP_CR.

Figure 297. AES-ECB mode encryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128: Key = [K3 K2].
If Key size = 192: Key = [K3 K2 K1]
If Key size = 256: Key = [K3 K2 K1 K0].

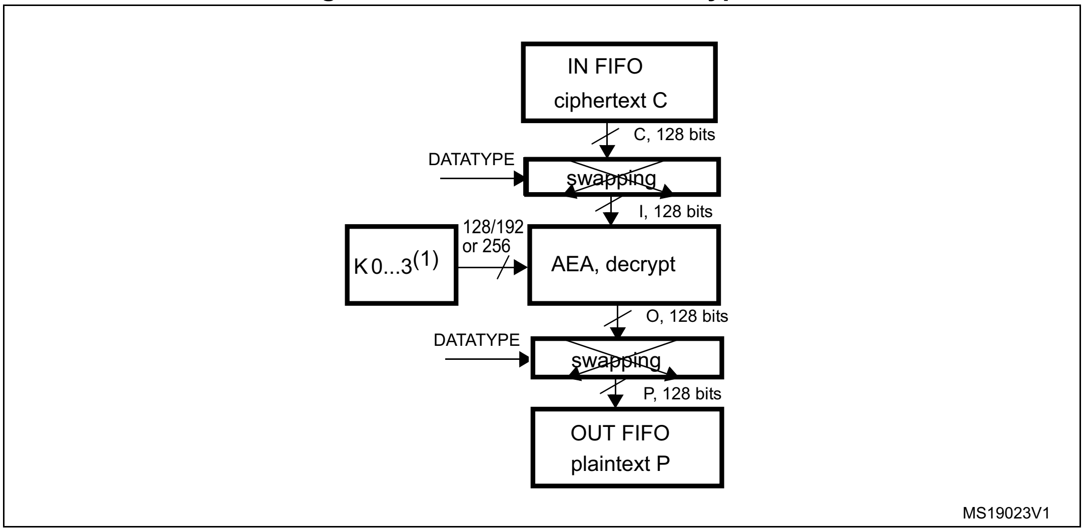
In this mode a 128- bit plaintext data block (P) is used after bit/byte/half-word swapping as the input block (I). The input block is processed through the AEA in the encrypt state using the 128, 192 or 256-bit key. The resultant 128-bit output block (O) is used after bit/byte/half-word swapping as ciphertext (C). It is then pushed into the OUT FIFO.

For more information on data swapping refer to *Section 36.4.16: CRYP data registers and data swapping*.

AES ECB decryption

Figure 298 illustrates the AES Electronic codebook (AES-ECB) mode decryption. This mode is selected by programming ALGOMODE to 0x4 and ALGODIR to 1 in CRYP_CR.

Figure 298. AES-ECB mode decryption



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.
2. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

To perform an AES decryption in ECB mode, the secret key has to be prepared (it is necessary to execute the complete key schedule for encryption) by collecting the last round key, and using it as the first round key for the decryption of the ciphertext. This preparation phase is computed by the AES core. Refer to *Section 36.4.7: Preparing the CRYP AES key for decryption* for more details on how to prepare the key.

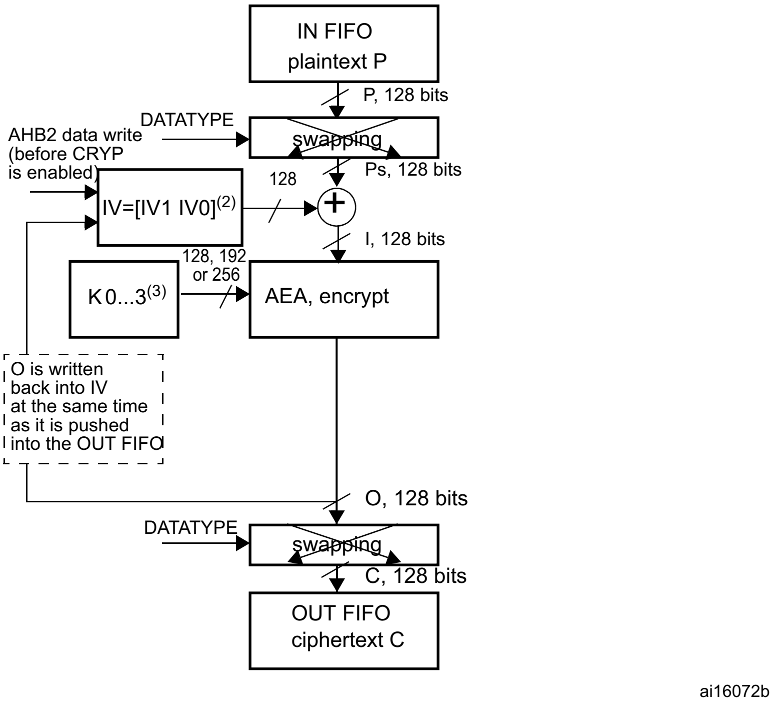
When the key preparation is complete, the decryption proceed as follows: a 128-bit ciphertext block (C) is used after bit/byte/half-word swapping as the input block (I). The keying sequence is reversed compared to that of the encryption process. The resultant 128-bit output block (O), after bit/byte or half-word swapping, produces the plaintext (P). The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

For more information on data swapping refer to *Section 36.4.16: CRYP data registers and data swapping*.

AES CBC encryption

Figure 299 illustrates the AES Cipher block chaining (AES-CBC) mode encryption. This mode is selected by writing ALGOMODE to 0x5 and ALGODIR to 0 in CRYP_CR.

Figure 299. AES-CBC mode encryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx = [IVxR IVxL], R = right, L = left.
3. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

In this mode the first input block (I_1) obtained after bit/byte/half-word swapping is formed by exclusive-ORing the first plaintext data block (P_1) with a 128-bit initialization vector IV ($I_1 = IV \oplus P_1$). The input block is processed through the AEA in the encrypt state using the 128-, 192- or 256-bit key (K0...K3). The resultant 128-bit output block (O_1) is used directly as ciphertext (C_1), that is, $C_1 = O_1$. This first ciphertext block is then exclusive-ORed with the second plaintext data block to produce the second input block, (I_2) = ($C_1 \oplus P_2$). Note that I_2 and P_2 now refer to the second block. The second input block is processed through the AEA to produce the second ciphertext block. This encryption process continues to “chain” successive cipher and plaintext blocks together until the last plaintext block in the message is encrypted.

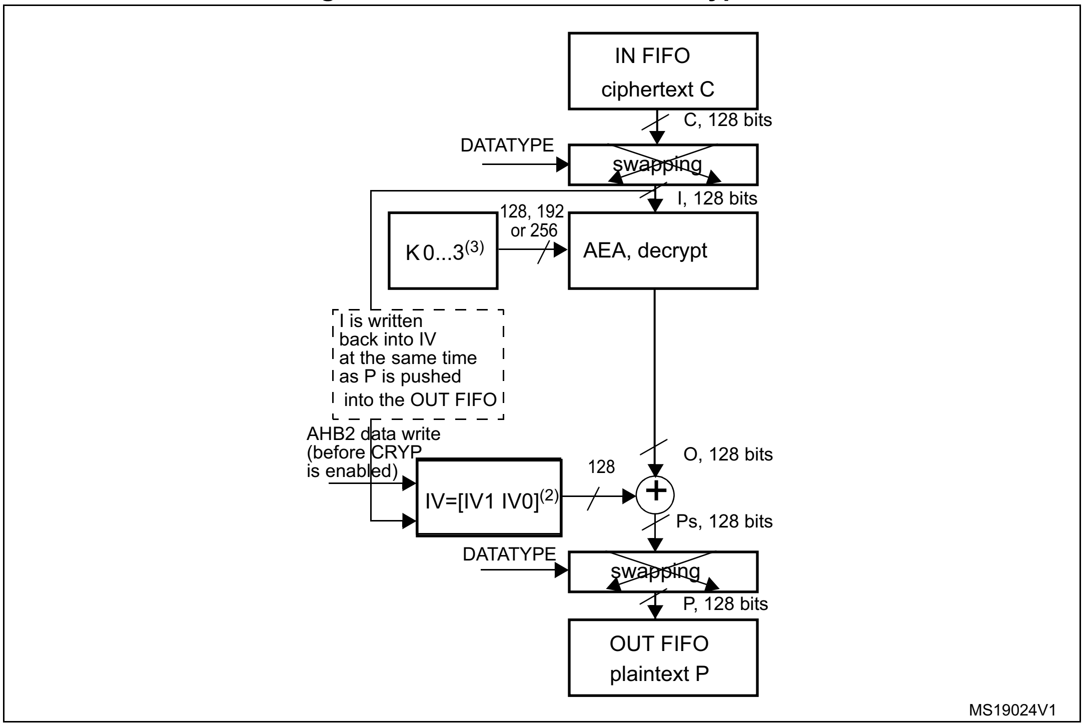
If the message does not consist of an integral number of data blocks, then the final partial data block should be encrypted in a manner specified for the application, as explained in *Section 36.4.8: CRYP stealing and data padding*.

For more information on data swapping, refer to *Section 36.4.16: CRYP data registers and data swapping*.

AES CBC decryption

Figure 300 illustrates the AES Cipher block chaining (AES-CBC) mode decryption. This mode is selected by writing ALGOMODE to 0x5 and ALGODIR to 1 in CRYP_CR.

Figure 300. AES-CBC mode decryption



1. K: key; C: cipher text; I: input block; O: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); P: plain text; IV: Initialization vectors.
2. IVx = [IVxR IVxL], R = right, L = left.
3. If Key size = 128 => Key = [K3 K2].
   If Key size = 192 => Key = [K3 K2 K1]
   If Key size = 256 => Key = [K3 K2 K1 K0].

In CBC mode, like in ECB mode, the secret key must be prepared to perform an AES decryption. Refer to *Section 36.4.7: Preparing the CRYP AES key for decryption* for more details on how to prepare the key.

When the key preparation process is complete, the decryption proceeds as follows: the first 128-bit ciphertext block (C_1) is used directly as the input block (I_1). The input block is processed through the AEA in the decrypt state using the 128-, 192- or 256-bit key. The resulting output block is exclusive-ORed with the 128-bit initialization vector IV (which must be the same as that used during encryption) to produce the first plaintext block ($P_1 = O_1 \oplus IV$).

The second ciphertext block is then used as the next input block and is processed through the AEA. The resulting output block is exclusive-ORed with the first ciphertext block to produce the second plaintext data block ($P_2 = O_2 \oplus C_1$). Note that P_2 and O_2 refer to the second block of data. The AES-CBC decryption process continues in this manner until the last complete ciphertext block has been decrypted.

Ciphertext representing a partial data block must be decrypted in a manner specified for the application, as explained in *Section 36.4.8: CRYP stealing and data padding*.

For more information on data swapping, refer to *Section 36.4.16: CRYP data registers and data swapping*.

AES suspend/resume operations in ECB/CBC modes

Before interrupting the current message, the user application must respect the following sequence:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM = 1 and OFNE = 0 in the CRYP_SR) and the BUSY bit is cleared.
3. If DMA is used, stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the CRYP by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2] in the CRYP_CR register). If ECB mode is not selected, save the initialization vector registers, because CRYP_IVx(L/R)R registers have changed from initial values during the data processing.

Note: Key registers do not need to be saved as the original key value is known by the application.

5. If DMA is used, save the DMA controller status (such as pointers to IN and OUT data transfers, number of remaining bytes).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be set to 0).
3. Configure the cryptographic processor again with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. For AES-ECB or AES-CBC decryption, the key must be prepared again, as described in *Section 36.4.7: Preparing the CRYP AES key for decryption*.
5. If ECB mode is not selected, restore CRYP_IVx(L/R)R registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again the DMA requests from the cryptographic processor, by setting DIEN and DOEN bits to 1 in the CRYP_DMACR register.

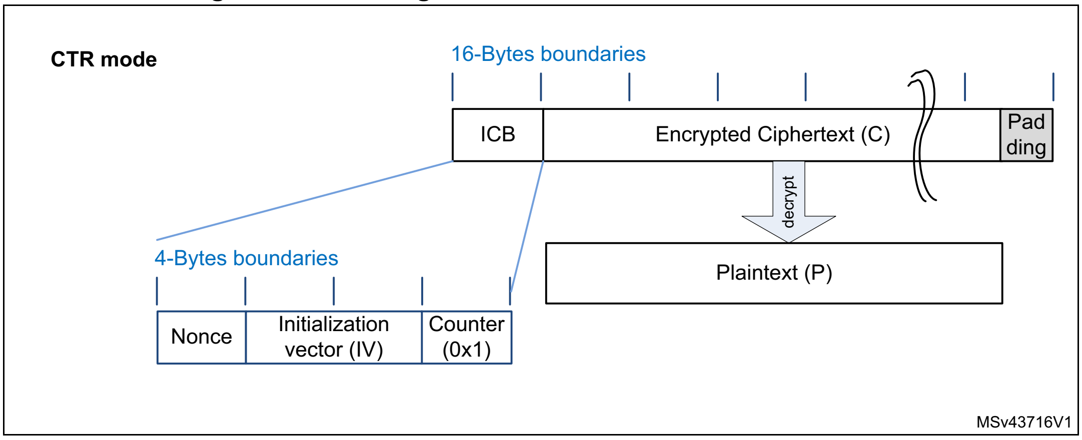
36.4.12 CRYP AES counter mode (AES-CTR)

Overview

The AES counter mode (CTR) uses the AES block as a key stream generator. The generated keys are then XORed with the plaintext to obtain the ciphertext.

CTR chaining is defined in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.* A typical message construction in CTR mode is given in *Figure 301*.

Figure 301. Message construction for the Counter mode



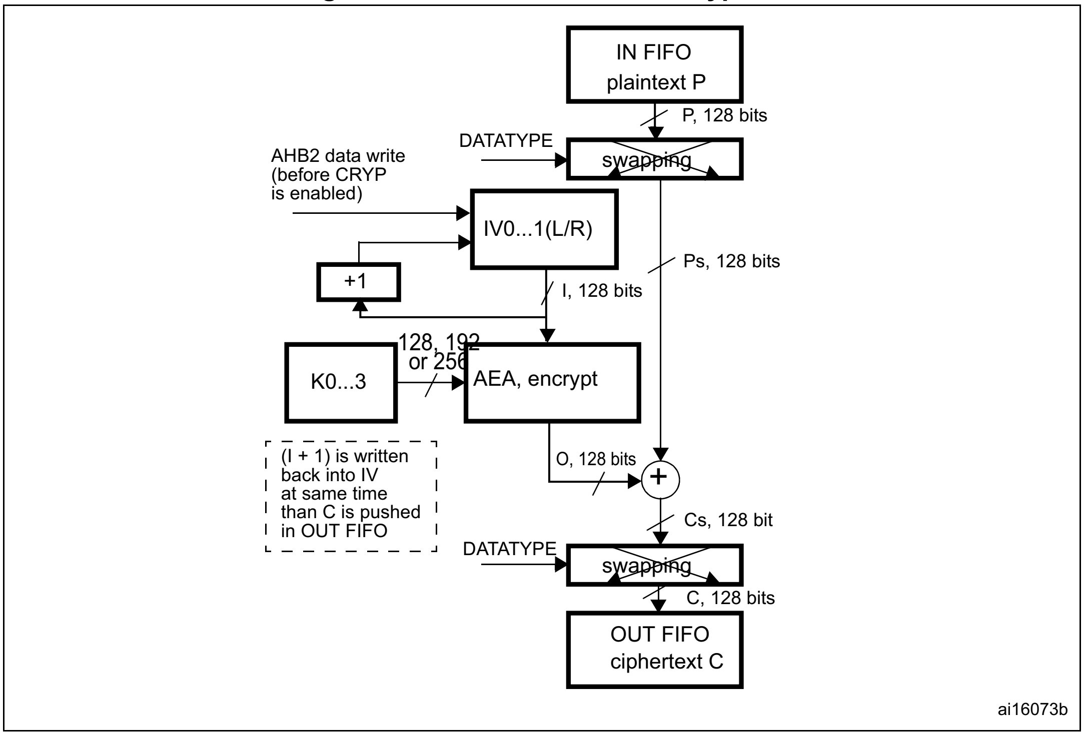
The structure of this message is as below:

- A 16-byte Initial Counter Block (ICB), composed of three distinct fields:
  - A *nonce:* a 32-bit, single-use value (i.e. a new nonce should be assigned to each new communication).
  - The *initialization vector* (IV): a 64-bit value that must be unique for each execution of the mode under a given key.
  - The *counter:* a 32-bit big-endian integer that is incremented each time a block has been processed. The initial value of the counter should be set to 1.
- The plaintext (P) is both authenticated and encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

AES CTR processing

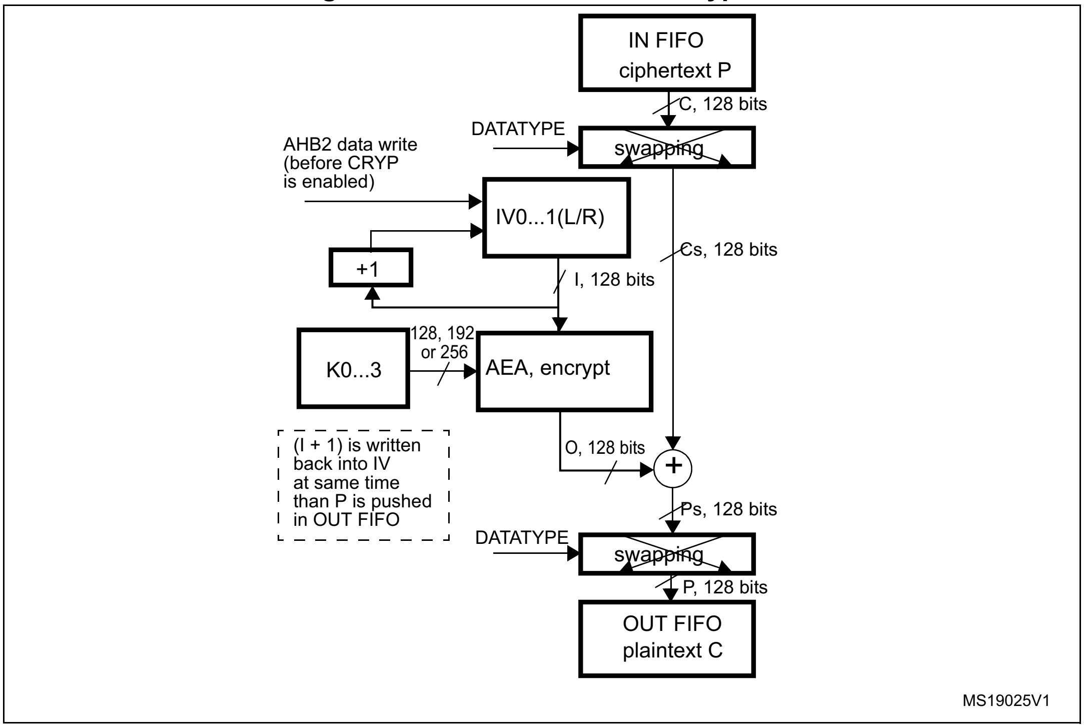
Figure 302 (respectively *Figure 303*) describes the AES-CTR encryption (respectively decryption) process implemented within this peripheral. This mode is selected by programming ALGOMODE bitfield to 0x6 in CRYP_CR.

Figure 302. AES-CTR mode encryption



1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

Figure 303. AES-CTR mode decryption



1. K: key; C: cipher text; I: input Block; o: output block; Ps: plain text before swapping (when decoding) or after swapping (when encoding); Cs: cipher text after swapping (when decoding) or before swapping (when encoding); P: plain text; IV: Initialization vectors.

In CTR mode, the output block is XORed with the subsequent input block before it is input to the algorithm. Initialization vectors in the peripheral must be initialized as shown on *Table 253*.

Table 253. Counter mode initialization vector

CRYP_IV1RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV0LR[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x1

Unlike in CBC mode, which uses the CRYP_IVx(L/R)R registers only once when processing the first data block, in CTR mode IV registers are used for processing each data block, and the peripheral increments the least significant 32 bits (leaving the other most significant 96 bits unchanged).

CTR decryption does not differ from CTR encryption, since the core always encrypts the current counter block to produce the key stream that will be XORed with the plaintext or cipher as input. Thus when ALGOMODE is set to 0x6, ALGODIR is don’t care.

Note: In this mode the key must NOT be prepared for decryption.

The following sequence must be used to perform an encryption or a decryption in CTR chaining mode:

1. Make sure the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.
2. Configure CRYP_CR as follows:
   a) Program ALGOMODE bits to 0x6 to select CTR mode. ALGODIR can be set to any value.
   b) Configure the data type (1, 8, 16 or 32 bits) through the DATATYPE bits.
   c) Define the key length using KEYSIZE bits.
3. Initialize the key registers (128,192 or 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) as described in *Table 253*.
4. Flush the IN and OUT FIFOs by writing the FFLUSH bit to 1 in the CRYP_CR register.
5. If it is the last block, optionally pad the data with zeros to have a complete block.
6. Append data in the cryptographic processor and read the result. The three possible scenarios are described in *Section 36.4.5: CRYP procedure to perform a cipher operation*.
7. Repeat the previous step until the second last block is processed. For the last block, execute the two previous steps. For this last block, the driver must discard the data that is not part of the data when the last block size is less than 16 bytes.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, and resume the message which was interrupted. Detailed CBC sequence can be found in *Section 36.4.11: CRYP AES basic chaining modes (ECB, CBC)*.

Note: Like for CBC mode, IV registers must be reloaded during the resume operation.

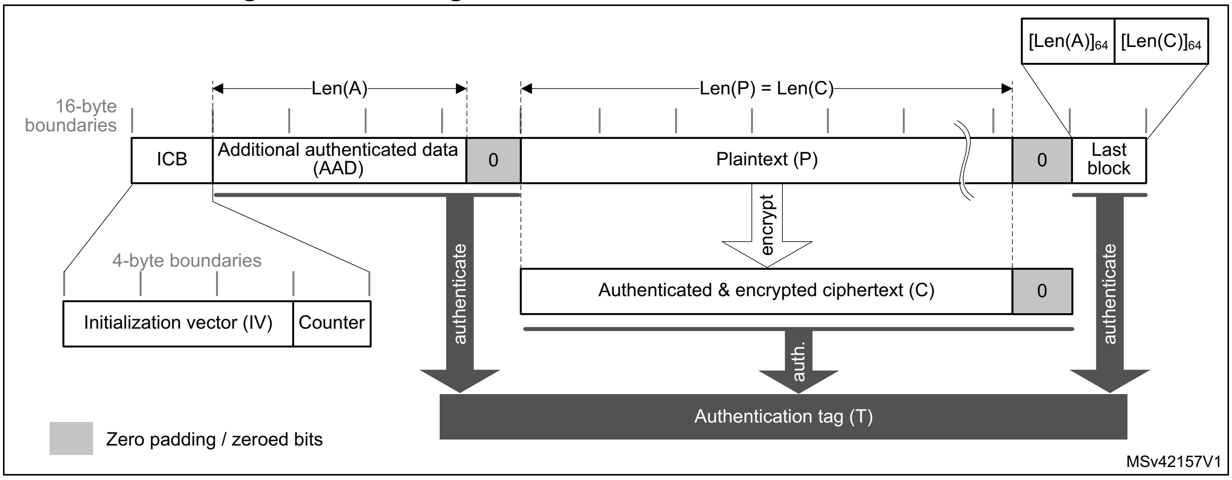
36.4.13 CRYP AES Galois/counter mode (GCM)

Overview

The AES Galois/counter mode (GCM) allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code). To ensure confidentiality, GCM algorithm is based on AES counter mode. It uses a multiplier over a fixed finite field to generate the tag.

GCM chaining is defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.* A typical message construction in GCM mode is given in *Figure 304*.

Figure 304. Message construction for the Galois/counter mode



The structure of this message is defined as below:

- A 16-byte Initial Counter Block (ICB), composed of two distinct fields:
  - The *initialization vector* (IV): a 96-bit value that must be unique for each execution of the mode under a given key. Note that the GCM standard supports IV that are shorter than 96-bit, but in this case strict rules apply.
  - The *counter:* a 32-bit big-endian integer that is incremented each time a block has been processed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- The authenticated header A (also knows as Additional Authentication Data) has a known length *Len(A)* that can be non-multiple of 16 bytes and cannot exceed 2^{64}-1 bits. This part of the message is only authenticated, not encrypted.
- The plaintext message (P) is both authenticated and encrypted as ciphertext C, with a known length *Len(P)* that can be non-multiple of 16 bytes, and cannot exceed 2^{32} -2 blocks of 128-bits.

Note: GCM standard specifies that ciphertext C has same bit length as the plaintext P.

- When a part of the message (AAD or P) has a length which is non-multiple of 16 bytes, a special padding scheme is required.
- The last block is composed of the length of A (on 64 bits) and the length of ciphertext C (on 64 bits) as shown in *Table 254*.

Table 254. GCM last block definition

Endianness	Bit[0]	Bit[32]	Bit[64]	Bit[96]
Input data	0x0	Header length[31:0]	0x0	Payload length[31:0]

AES GCM processing

This mode is selected by writing ALGOMODE bitfield to 0x8 in CRYP_CR.

The mechanism for the confidentiality of the plaintext in GCM mode is a variation of the Counter mode, with a particular 32-bit incrementing function that generates the necessary sequence of counter blocks.

CRYP_IVx(L/R)R registers are used for processing each data block. The cryptographic processor automatically increments the 32 least signification bits of the counter block. The first counter block (CB1) written by the application is equal to the Initial Counter Block incremented by one (see *Table 255*).

Table 255. GCM mode IV registers initialization

Register	CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
Input data	ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x2

Note: In this mode the key must NOT be prepared for decryption.

The authentication mechanism in GCM mode is based on a hash function, called *GF2mul*, that performs multiplication by a fixed parameter, called the hash subkey (H), within a binary Galois field.

To process a GCM message, the driver must go through four phases, which are described in the following subsections.

- The Init phase: the peripheral prepares the GCM hash subkey (H) and performs the IV processing
- The Header phase: the peripheral processes the Additional Authenticated Data (AAD), with hash computation only.
- The Payload phase: the peripheral processes the plaintext (P) with hash computation, keystream encryption and data XORing. It operates in a similar way for ciphertext (C).
- The Final phase: the peripheral generates the authenticated tag (T) using the data last block.

1. **GCM initialization phase**

   During this first step, the GCM hash subkey (H) is calculated and saved internally to be used for processing all the blocks. It is recommended to follow the sequence below:

   a) Make sure the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.

   b) Select the GCM chaining mode by programming ALGOMODE bits to 0x8 in CRYP_CR. In the same register define the key length using KEYSIZE bits and the data type using DATATYPE bits.

   c) Configure GCM_CCMPH bits to 0b00 in CRYP_CR to indicate that the initialization phase is ongoing.

   d) Initialize the key registers (128, 192 or 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) as defined in *Table 255*.

   e) Set CRYPEN bit to 1 to start the calculation of the hash key.

   f) Wait for the CRYPEN bit to be cleared to 0 by the cryptographic processor, before moving on to the next phase.

2. **GCM header phase**

   The below sequence shall be performed after the GCM initialization phase. It must be complete before jumping to the payload phase. The sequence is identical for encryption and decryption.

   g) Set the GCM_CCMPH bits to 0b01 in CRYP_CR to indicate that the header phase is ongoing.

   h) Set the CRYPEN bit to 1 to start accepting data.

   i) If it is the last block of additional authenticated data, optionally pad the data with zeros to have a complete block.

   j) Append additional authenticated data in the cryptographic processor. The three possible scenarios are described in *Section 36.4.5: CRYP procedure to perform a cipher operation*.

   k) Repeat the previous step until the second last additional authenticated data block is processed. For the last block, execute the two previous steps. Once all the additional authenticated data have been supplied, wait until the BUSY flag is cleared before moving on to the next phase.

Note: This phase can be skipped if there is no additional authenticated data, i.e. Len(A) = 0.

In header and payload phases, CRYPEN bit is not automatically cleared by the cryptographic processor.

3. **GCM payload phase (encryption or decryption)**

When the payload size is not null, this sequence must be executed after the GCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

l) Set the CRYPEN bit to 0.

m) Configure GCM_CCMPH to 0b10 in the CRYP_CR register to indicate that the payload phase is ongoing.

n) Select the algorithm direction (0 for encryption, 1 for decryption) through the ALGODIR bit in CRYP_CR.

o) Set the CRYPEN bit to 1 to start accepting data.

p) If it is the last block of cleartext or plaintext, optionally pad the data with zeros to have a complete block. For encryption, refer to *Section 36.4.8: CRYP stealing and data padding* for more details.

q) Append payload data in the cryptographic processor, and read the result. The three possible scenarios are described in *Section 36.4.5: CRYP procedure to perform a cipher operation*.

r) Repeat the previous step until the second last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block, the driver must discard the bits that are not part of the cleartext or the ciphertext when the last block size is less than 16 bytes. Once all payload data have been supplied, wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload data, i.e. Len(C) = 0 (see GMAC mode).

4. **GCM final phase**

In this last step, the cryptographic processor generates the GCM authentication tag and stores it in CRYP_DOUT register.

s) Configure GCM_CCMPH[1:0] to 0b11 in CRYP_CR to indicate that the Final phase is ongoing. Set the ALGODIR bit to 0 in the same register.

t) Write the input to the CRYP_DIN register four times. The input must be composed of the length in bits of the additional authenticated data (coded on 64 bits) concatenated with the length in bits of the payload (coded of 64 bits), as show in *Table 254*.

Note: In this final phase, data have to be inserted normally (no swapping).

u) Wait until the OFNE flag (FIFO output not empty) is set to 1 in the CRYP_SR register.

v) Read the CRYP_DOUT register four times: the output corresponds to the authentication tag.

w) Disable the cryptographic processor (CRYPEN bit = 0 in CRYP_CR)

x) If an authenticated decryption is being performed, compare the generated tag with the expected tag passed with the message.

Suspend/resume operations in GCM mode

Before interrupting the current message in header or payload phase, the user application must respect the following sequence:

1. If DMA is used, stop DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM = 1 and OFNE = 0 in the CRYP_SR register) and the BUSY bit is cleared.
3. If DMA is used, stop DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2], bits [17:16] and bits 19 of the CRYP_CR register). In addition, save the initialization vector registers, since CRYP_IVx(L/R)R registers have changed from their initial values during data processing.

Note: Key registers do not need to be saved as original their key value is known by the application.

5. Save context swap registers: CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R
6. If DMA is used, save the DMA controller status (pointers to IN and OUT data transfers, number of remaining bytes, etc.).

To resume message processing, the user must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (it must be 0).
3. Configure again the cryptographic processor with the initial setting in CRYP_CR, as well as the key registers using the saved configuration.
4. Restore context swap registers: CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R
5. Restore CRYP_IVx(L/R)R registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again cryptographic processor DMA requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Note: In Header phase, DMA OUT FIFO transfer is not used.

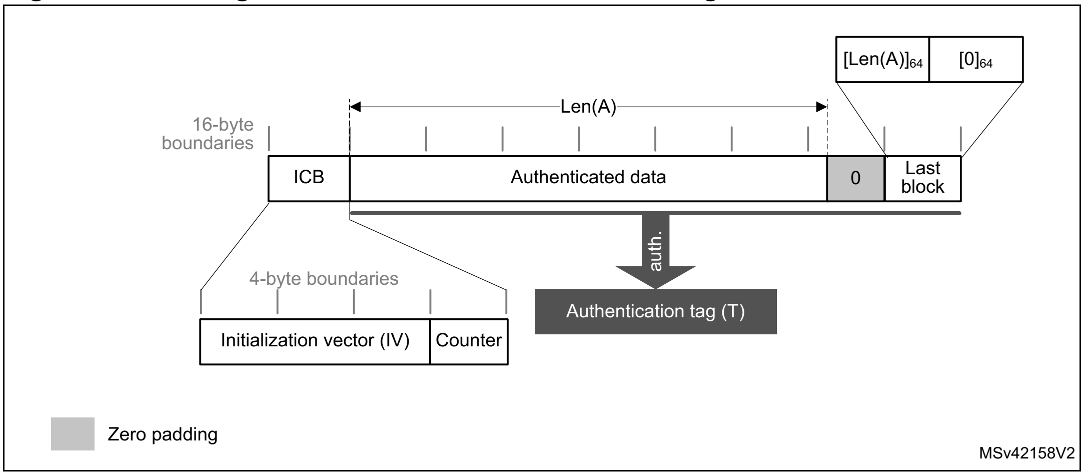
36.4.14 CRYP AES Galois message authentication code (GMAC)

Overview

The Galois message authentication code (GMAC) allows authenticating a plaintext and generating the corresponding tag information (also known as message authentication code). It is based on GCM algorithm, as defined in NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

A typical message construction in GMAC mode is given in *Figure 305*.

Figure 305. Message construction for the Galois Message Authentication Code mode



AES GMAC processing

This mode is selected by writing ALGOMODE bitfield to 0x8 in CRYP_CR.

GMAC algorithm corresponds to the GCM algorithm applied on a message composed only of an header. As a consequence, all steps and settings are the same as in GCM mode, except that the payload phase (3) is not used.

Suspend/resume operations in GMAC

GMAC is exactly the same as GCM algorithm except that only header phase (2) can be interrupted.

36.4.15 CRYP AES Counter with CBC-MAC (CCM)

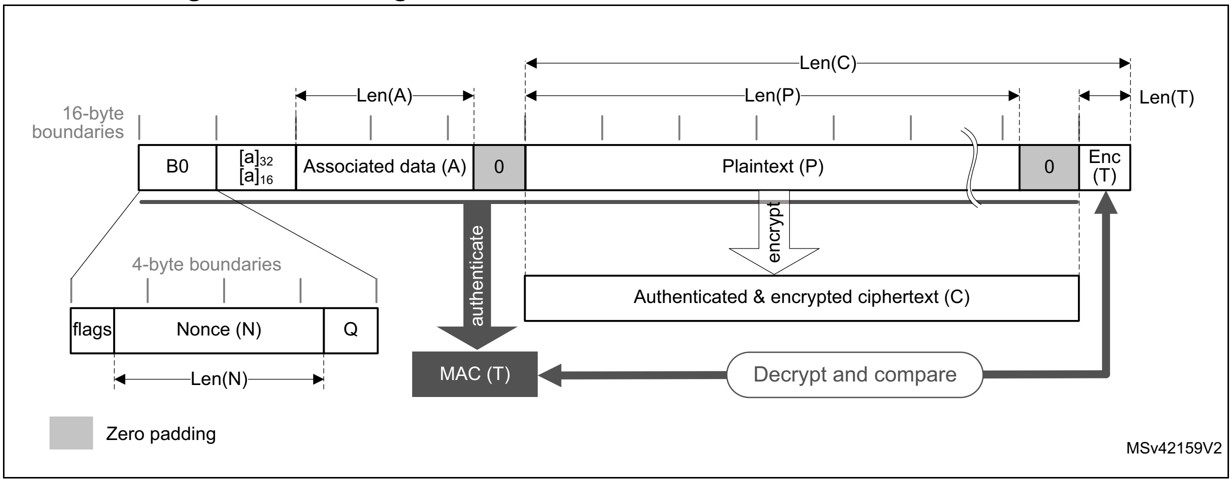
Overview

The AES Counter with Cipher Block Chaining-Message Authentication Code (CCM) algorithm allows encrypting and authenticating the plaintext, and generating the correspondent ciphertext and tag (also known as message authentication code). To ensure confidentiality, CCM algorithm is based on AES counter mode. It uses Cipher Block Chaining technique to generate the message authentication code. This is commonly called CBC-MAC

Note: NIST does not approve this CBC-MAC as an authentication mode outside of the context of the CCM specification.

CCM chaining is specified in NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. A typical message construction in CCM mode is given in *Figure 306*

Figure 306. Message construction for the Counter with CBC-MAC mode



The structure of this message is as below:

- One 16-byte first authentication block (called B0 by the standard), composed of three distinct fields:
  - *Q*: a bit string representation of the byte length of P (Plen)
  - A *nonce* (N)*:* single-use value (i.e. a new nonce should be assigned to each new communication). Size of nonce *Nlen* + size of *Plen* shall be equal to 15 bytes.
  - Flags: most significant byte containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values t (MAC length expressed in bytes) and q (plaintext length such as Plen<2^{8q} bytes). Note that the counter blocks range associated to q is equal to 2^{8q-4}, i.e. if q maximum value is 8, the counter blocks used in cipher shall be on 60 bits.

- 16-bytes blocks (B) associated to the Associated Data (A).

  This part of the message is only authenticated, not encrypted. This section has a known length, *ALen,* that can be a non-multiple of 16 bytes (see *Figure 306*). The standard also states that, on the MSB bits of the first message block (B1), the associated data length expressed in bytes (*a*) must be encoded as defined below:
  – If $0 < a < 2^{16}-2^{8}$, then it is encoded as $[a]_{16}$, i.e. two bytes.
  – If $2^{16}-2^{8} < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, i.e. six bytes.
  – If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, i.e. ten bytes.
- 16-byte blocks (B) associated to the plaintext message (P), which is both authenticated and encrypted as ciphertext C, with a known length of *Plen*. This length can be a non-multiple of 16 bytes (see *Figure 306*).
- The encrypted MAC (T) of length *Tlen* appended to the ciphertext C of overall length *Clen*.
- When a part of the message (A or P) has a length which is a non-multiple of 16 bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (i.e. no payload).

As an example, the C.1 section in *NIST Special Publication 800-38C* gives the following:

`N:` **`10111213 141516`** (Nlen = 56 bits or 0x7 bytes)

`A:` ***`00010203 04050607`*** (Alen = 64 bits or 0x8 bytes)

`P:` *`20212223`* (Plen = 32 bits i.e. Q = 0x*4* bytes)

`T: 6084341b` (Tlen = 32 bits or t = 4)

```
B0: 4f101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000
```

```
CTR0: 0710111213 141516 00000000 00000000
CTR1: 0710111213 141516 00000000 00000001
```

The usage of control blocks CTRx is explained in the following section. The generation of CTR0 from the first block (B0) must be managed by software.

AES CCM processing

This mode is selected by writing ALGOMODE bitfield to 0x9 in CRYP_CR.

The data input to the generation-encryption process are a valid nonce, a valid payload string, and a valid associated data string, all properly formatted. The CBC chaining mechanism is applied to the formatted data to generate a MAC, whose length is known. Counter mode encryption, which requires a sufficiently long sequence of counter blocks as input, is applied to the payload string and separately to the MAC. The resulting data, called the ciphertext C, is the output of the generation-encryption process on plaintext P.

CRYP_IVx(L/R)R registers are used for processing each data block. The cryptographic processor automatically increments the CTR counter with a bit length defined by the first block (B0). The first counter written by application, CTR1, is equal to B0 with the first 5 bits zeroed and the most significant bits containing P byte length also zeroed, then incremented by one (see *Table 256*).

Table 256. CCM mode IV registers initialization

Register	CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
Input data	B0[127:96], where the 5 most significant bits are set to 0 (flag bits)	B0[95:64]	B0[63:32]	B0[31:0], where Q length bits are set to 0, except for bit 0 that is set to 1

Note: In this mode, the key must NOT be prepared for decryption.

To process a CCM message, the driver must go through four phases, which are described below.

- The **Initialization phase**: the peripheral processes the first block and prepares the first counter block.
- The **Header phase**: the peripheral processes the Associated data (A), with hash computation only.
- The **Payload phase**: the peripheral processes the plaintext (P), with hash computation, counter block encryption and data XORing. It operates in a similar way for ciphertext (C).
- The **Final phase**: the peripheral generates the message authentication code (MAC).

1. **CCM initialization phase**

   In this first step, the first block (B0) of the CCM message is programmed into the CRYP_DIN register. During this phase, the CRYP_DOUT register does not contain any output data. It is recommended to follow the sequence below:

   a) Make sure that the cryptographic processor is disabled by clearing the CRYPEN bit in the CRYP_CR register.

   b) Select the CCM chaining mode by programming the ALGOMODE bits to 0x9 in the CRYP_CR register. In the same register define the key length using KEYSIZE bits and the data type using DATATYPE bits.

   c) Configure the GCM_CCMPH bits to 0b00 in CRYP_CR to indicate that we are in the initialization phase.

   d) Initialize the key registers (128, 192 or 256 bits) in CRYP_KEYRx as well as the initialization vector (IV) with CTR1 information, as defined in *Table 256*.

   e) Set the CRYPEN bit to 1 in CRYP_CR to start accepting data.

   f) Write the B0 packet into CRYP_DIN register, then wait for the CRYPEN bit to be cleared to 0 by the cryptographic processor before moving on to the next phase.

Note: In this initialization phase, data have to be inserted normally (no swapping).

2. **CCM header phase**

   The below sequence shall be performed after the CCM initialization phase. It must be complete before jumping to the payload phase. The sequence is identical for encryption and decryption. During this phase, the CRYP_DOUT register does not contain any output data.

   g) Set the GCM_CCMPH bit to 0b01 in CRYP_CR to indicate that the header phase is ongoing.

   h) Set the CRYPEN bit to 1 to start accepting data.

   i) If it is the last block of associated data, optionally pad the data with zeros to have a complete block.

   j) Append the associated data in the cryptographic processor. The three possible scenarios are described in *Section 36.4.5: CRYP procedure to perform a cipher operation*.

   k) Repeat the previous step until the second last associated data block is processed. For the last block, execute the two previous steps. Once all the additional authenticated data have been supplied, wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no associated data (Alen = 0).

The first block of the associated data B1 must be formatted with the associated data length. This task must be managed by the driver.

3. **CCM payload phase (encryption or decryption)**

When the payload size is not null, this sequence must be performed after the CCM header phase. During this phase, the encrypted/decrypted payload is stored in the CRYP_DOUT register.

l) Set the CRYPEN bit to 0.

m) Configure GCM_CCMPH bits to 0b10 in CRYP_CR to indicate that the payload phase is ongoing.

n) Select the algorithm direction (0 for encryption, 1 for decryption) through the ALGODIR bit in CRYP_CR.

o) Set the CRYPEN bit to 1 to start accepting data.

p) If it is the last block of cleartext, optionally pad the data with zeros to have a complete block (encryption only). For decryption, refer to *Section 36.4.8: CRYP stealing and data padding* for more details.

q) Append payload data in the cryptographic processor, and read the result. The three possible scenarios are described in *Section 36.4.5: CRYP procedure to perform a cipher operation*.

r) Repeat the previous step until the second last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), execute the two previous steps. For the last block of ciphertext (decryption only), the driver must discard the data that is not part of the cleartext when the last block size is less than 16 bytes. Once all payload data have been supplied, wait until the BUSY flag is cleared

Note: This phase can be skipped if there is no payload data, i.e. Plen = 0 or Clen = Tlen

Note: Do not forget to remove LSB_{Tlen}(C) encrypted tag information when decrypting ciphertext C.

4. **CCM final phase**

In this last step, the cryptographic processor generates the CCM authentication tag and stores it in the CRYP_DOUT register.

s) Configure GCM_CCMPH[1:0] bits to 0b11 in CRYP_CR to indicate that the final phase is ongoing and set the ALGODIR bit to 0 in the same register.

t) Load in CRYP_DIN, the CTR0 information which is described in *Table 256* with bit[0] set to 0.

Note: In this final phase, data have to be inserted normally (no swapping).

u) Wait until the OFNE flag (FIFO output not empty) is set to 1 in the CRYP_SR register.

v) Read the CRYP_DOUT register four times: the output corresponds to the encrypted CCM tag.

w) Disable the cryptographic processor (CRYPEN bit set to 0 in CRYP_CR)

x) If an authenticated decryption is being performed, compare the generated encrypted tag with the encrypted tag padded in the ciphertext, i.e. LSB_{Tlen}(C) = MSB_{Tlen}(CRYP_DOUT data).

Suspend/resume operations in CCM mode

Before interrupting the current message in payload phase, the user application must respect the following sequence:

1. If DMA is used, stop the DMA transfers to the IN FIFO by clearing to 0 the DIEN bit in the CRYP_DMACR register.
2. Wait until both the IN and the OUT FIFOs are empty (IFEM = 1 and OFNE = 0 in the CRYP_SR register) and the BUSY bit is cleared.
3. If DMA is used, stop the DMA transfers from the OUT FIFO by clearing to 0 the DOEN bit in the CRYP_DMACR register.
4. Disable the cryptographic processor by setting the CRYPEN bit to 0 in CRYP_CR, then save the current configuration (bits [9:2], bits [17:16] and bits 19 in the CRYP_CR register). In addition, save the initialization vector registers, since CRYP_IVx(L/R)R registers have changed from their initial values during the data processing.

Note: Key registers do not need to be saved as their original key value is known by the application.

5. Save context swap registers: CRYP_CSGCMCCM0..7R
6. If DMA is used, save the DMA controller status (pointers for IN and OUT data transfers, number of remaining bytes, etc.).

To resume message processing, the user application must respect the following sequence:

1. If DMA is used, reconfigure the DMA controller to complete the rest of the FIFO IN and FIFO OUT transfers.
2. Make sure the cryptographic processor is disabled by reading the CRYPEN bit in CRYP_CR (must be 0).
3. Configure the cryptographic processor again with the initial setting in CRYP_CR and key registers using the saved configuration.
4. Restore context swap registers: CRYP_CSGCMCCM0..7R
5. Restore CRYP_IVx(L/R)R registers using the saved configuration.
6. Enable the cryptographic processor by setting the CRYPEN bit to 1.
7. If DMA is used, enable again cryptographic processor DMA requests by setting to 1 the DIEN and DOEN bits in the CRYP_DMACR register.

Note: In Header phase DMA OUT FIFO transfer is not used.

36.4.16 CRYP data registers and data swapping

The CRYP_DIN register is the 32-bit wide data input register of the peripheral. It is used to enter into the input FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

The four adjacent (respectively two) words of the AES (respectively DES/TDES) data block are organized in big-endian order, with the most significant byte of a word on the lowest address.

The cryptographic accelerator can be configured to perform a bit, byte, half-word, or no swapping on the input data word in the CRYP_DINR register, before loading it to the CRYP processing core, and on the data output from the CRYP processing core, before sending it to the CRYP_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0] bitfield of the AES_CR register. The selection applies both to the input and the output of the CRYP processing core.

Note: The CRYP Key registers (CRYP_Kx(L/R)) and initialization registers (CRYP_IVx(L/R)) are not sensitive to the swap mode selected.

More information on data input and data swapping can be found in the next subsections.

DES/TDES data input and output

A 64-bit data block is entered into the cryptographic processor with two successive 32-bit word write operations to the CRYP_DINR register (DIN[31:0] bitfield), the most significant word (bits [64:33]) first, the least significant word (bits [32:1]) last.

A 64-bit data block is retrieved from the cryptographic processor with two successive 32-bit word read operations from the CRYP_DOUTR register (DOUT[31:0] bitfield), the most significant word (bits [64:33]) first, the least significant word (bits [32:1]) last.

DES/TDES data swapping feature

The cryptographic processor data swapping feature for DES/TDES is summarized in *Table 257* and *Figure 307*.

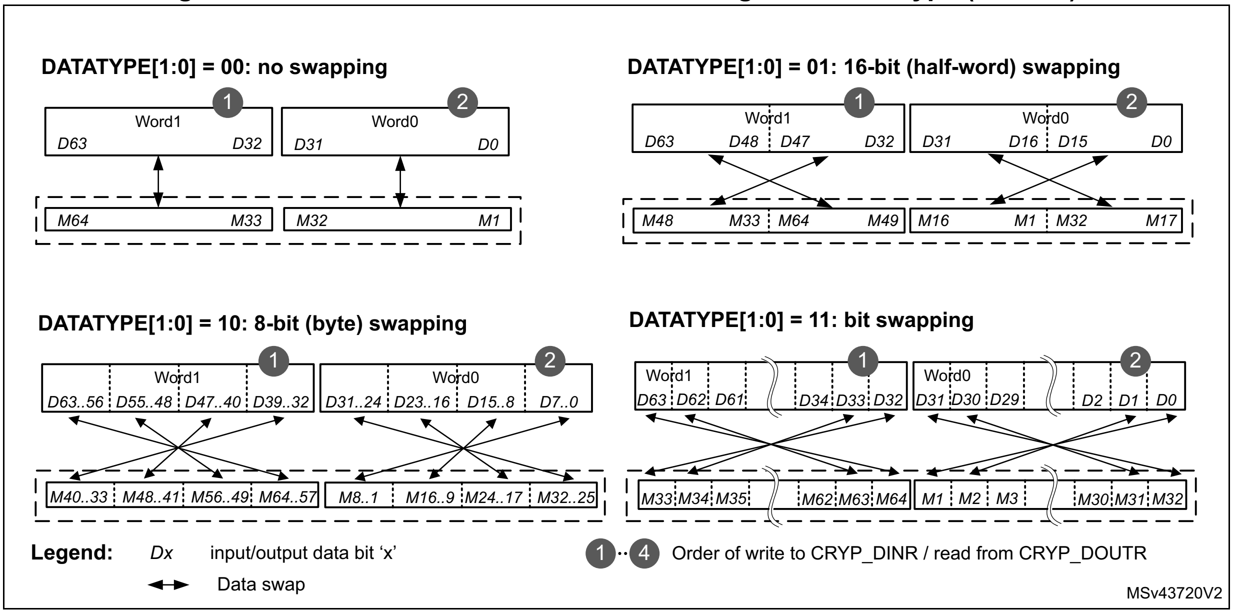
Table 257. DES/TDES data swapping example

DATATYPE in CRYP_CR	Swapping performed	Data block representation (64-bit) / System memory data (big-endian)
00	No swapping	Block[64..1]: 0xABCD7720 6973FE01
		Address @, word[63..32]: 0xABCD 7720 Address @+4, word[31..0]: 0x6973 FE01
01	Half-word (16-bit) swapping	Block[64..1]: 0x**ABCD** 7720 **6973** FE01
		Address @, word[63..32]: 0x7720 **ABCD** Address @+4, word[31..0]: 0xFE01 **6973**
10	Byte (8-bit) swapping	Block[64..1]: 0x**AB** CD **77** 20 **69** 73 **FE** 01
		Address @, word[63..32]: 0x20**77** CD**AB** Address @+4, word[31..0]: 0x01**FE** 73**69**

11	Bit swapping in 32-bit word	Block[64..33]: 0xABCD7720
		`1010 1011 1100 1101 0111 0111 0010 0000`
		Block[32..1]: 0x6973FE01
		`0110 1001 0111 0011 1111 1110 0000 0001`

		Address @, word[63..32]: 0x04EEB3D5
		`0000 0100 1110 1110 1011 0011 1101 0101`
		Address @+4, word[31..0]: 0x807FCE96
		`1000 0000 0111 1111 1100 1110 1001 0110`

Figure 307. 64-bit block construction according to the data type (IN FIFO)



AES data input and output

A 128-bit data block is entered into the cryptographic processor with four successive 32-bit word writes into the CRYP_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 128-bit data block is retrieved from the cryptographic processor with four successive 32-bit word reads from the CRYP_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

AES data swapping feature

The cryptographic processor data swapping feature for AES is summarized in *Table 258* and *Figure 308*.

Note: Data swapping does not apply to GCM and CCM final phases. Data can be inserted normally by the application.

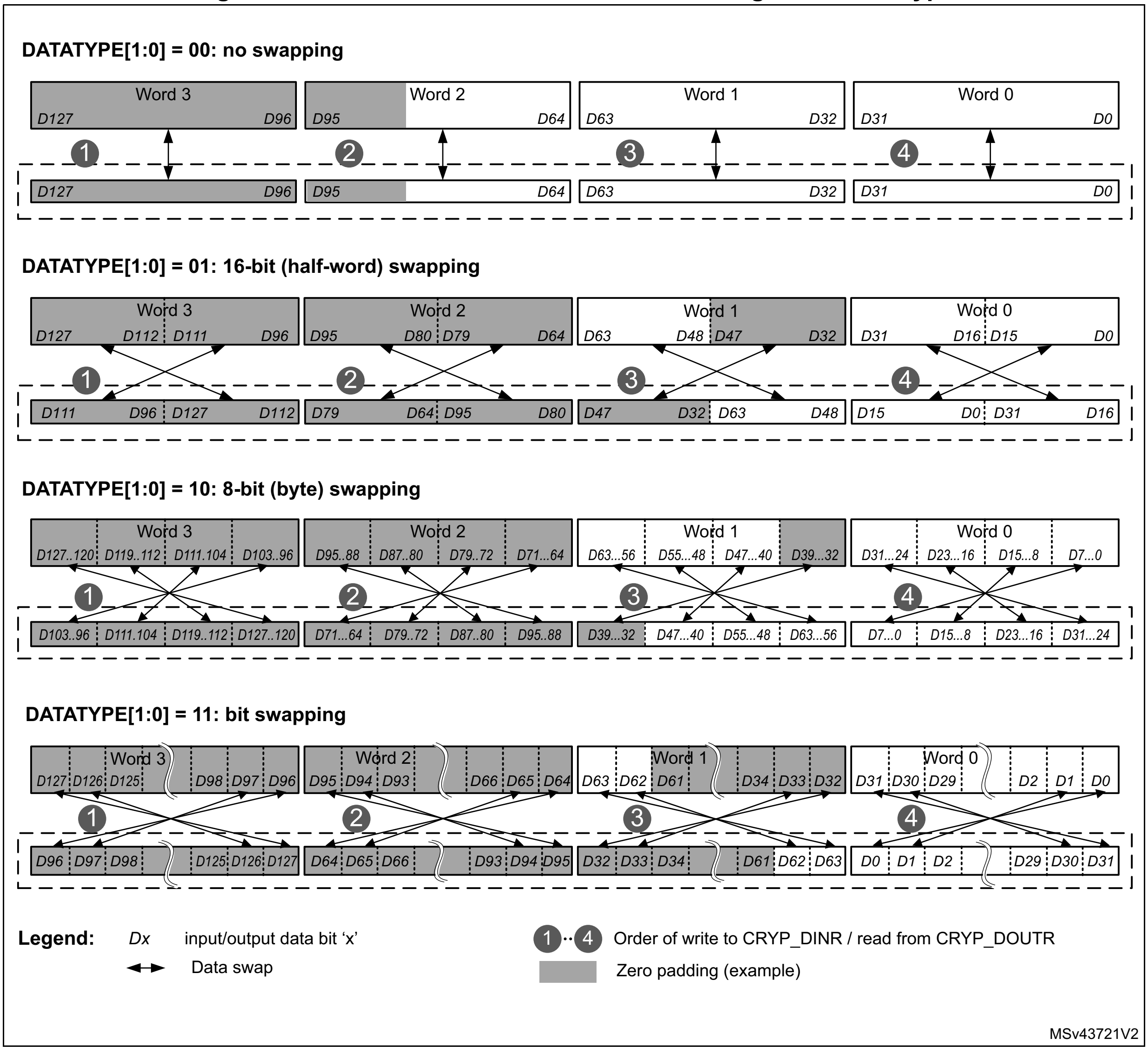
Table 258. AES data swapping example

DATATYPE in CRYP_CR	Swapping performed	First half data block (64-bit)
		System memory data (big-endian)
00	No swapping	Block[63..0]: 0x4E6F7720 69732074
		Address @, word[63..32]: 0x4E6F7720 Address @+4, word[31..0]: 0x69732074
01	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @+4, word[31..0]: 0x2074 **6973**

Table 258. AES data swapping example (continued)

DATATYPE in CRYP_CR	Swapping performed	First half data block (64-bit) / System memory data (big-endian)
10	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @+4, word[31..0]: 0x74**20** 73**69**
11	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @+4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 308. 128-bit block construction according to the data type



36.4.17 CRYP key registers

The CRYP_Kx registers are write-only registers used to store the encryption or decryption keys. They are organized as four 64-bit registers, as shown in *Table 259* and *Table 260*.

Note: In memory and in CRYP key registers, AES and DES/TDES keys are stored in big-endian format, with most significant byte on the lowest address.

Table 259. Key endianness in CRYP_KxR/LR registers (AES 128/192/256-bit keys)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	-	-	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
-	-	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]

Table 259. Key endianness in CRYP_KxR/LR registers (AES 128/192/256-bit keys) (continued)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
k[255:224]	k[223:192]	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]

Table 260. Key endianness in CRYP_KxR/LR registers (DES K1 and TDES K1/2/3)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	K1[64:33]	K1[32:1]	-	-	-	-
-	-	K1[64:33]	K1[32:1]	K2[64:33]	K2[32:1]	K3[64:33]	K3[32:1]

As shown on *Table 260*, when TDES is selected (ALGOMODE[2:0] = 000 or 001) a 3-key vector (K1, K2, and K3) is used. When DES is selected (ALGOMODE[2:0] = 010 or 011) only 1-key vector (K1) is used.

Note: DES/TDES keys include 8-bit parity information that are not used by the cryptographic processor. In other words, bits 8, 16, 24, 32, 40, 48, 56 and 64 of each 64-bit key value Kx[1:64] are not used.

Write operations to the CRYP_Kx(L/R) registers when the BUSY bit is set to 1 in the CRYP_SR register are ignored (which means that the register content is not modified). The application must thus check that the BUSY bit is cleared to 0 before modifying key registers.

Key registers are not affected by the data swapping controlled by DATATYPE value in CRYP_CR register.

Refer to *Section 36.7: CRYP registers* for a detailed description of CRYP_Kx(L/R) registers.

36.4.18 CRYP initialization vector registers

The CRYP_IVxL/IVxR registers are used to store the initialization vector or the nonce, depending on the chaining mode selected. When used, these registers are updated by the core after each computation round of the TDES or AES core.

They are organized as four 64-bit registers, as shown in shown in *Table 261* and *Table 262*. In DES/TDES mode only CRYP_IV0x are used.

Note: In memory and in CRYP IV registers, AES and DES/TDES initialization vectors *are stored in big-endian format, with most significant byte on the lowest address.*

Table 261. Initialization vector endianness in CRYP_IVx(L/R)R registers (AES)

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Table 262. Initialization vector endianness in CRYP_IVx(L/R)R registers (DES/TDES)

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
IVI[64:32]	IVI[31:0]	-	-

Any write operation to the CRYP_IV0...1(L/R)R registers when the BUSY bit is set to 1 in the CRYP_SR register is disregarded (which means that register content not modified). The software must thus check that the BUSY bit is cleared to 0 in the CRYP_SR register before modifying initialization vectors.

Reading the CRYP_IV0...1(L/R)R register returns the latest counter value (useful for managing suspend mode) except for CCM/GCM.

The initialization vector registers are not affected by the data swapping feature controlled by DATATYPE value in CRYP_CR register.

Refer to *Section 36.7: CRYP registers* for a detailed description of CRYP_IVxL/IVxR registers.

36.4.19 CRYP DMA interface

The cryptographic processor provides an interface to connect to the DMA (Direct Memory Access) controller. The DMA operation is controlled through the CRYP DMA control register (CRYP_DMACR).

Note: DMA cannot be used when DES/TDES algorithms are selected.

Data input using DMA

DMA can be enabled for writing data into the cryptographic peripheral by setting the DIEN bit in the CRYP_DMACR register. When this bit is set, the cryptographic processor initiates a DMA request during the INPUT phase each time it requires a word to be written to the CRYP_DIN register.

Table 263 shows the recommended configuration to transfer data from memory to cryptographic processor through the DMA controller.

Table 263. Cryptographic processor configuration for memory-to-peripheral DMA transfers

DMA channel control register field	Programming recommendation
Transfer size	Message length, multiple of AES cryptographic block size (4 words). *Note: DES/TDES is not supported with DMA.* According to the mode selected, special padding/ ciphertext stealing might be required. As an example, in case of AES GCM encryption or AES CCM decryption, DMA transfers must not include the last block. Refer to *Section 36.4.5: CRYP procedure to perform a cipher operation* for details.
Source burst size (memory)	If high latency memory is used (DDR): CRYP FIFO_size x2 /transfer_width = **16** (FIFO_size = 8x32-bit, transfer_width = 32-bit) If low latency memory is used (SRAM): CRYP FIFO_size /2 /transfer_width = **4**
Destination burst size (peripheral)	CRYP FIFO_size /2 /transfer_width = **4** (FIFO_size = 8x32-bit, transfer_width = 32-bit)
DMA FIFO size	If high latency memory is used (DDR): CRYP FIFO_size x2 = 64 bytes If low latency memory is used (SRAM): CRYP FIFO_size /2 = 16 bytes
Source transfer width (memory)	32-bit words

Table 263. Cryptographic processor configuration for memory-to-peripheral DMA transfers (continued)

DMA channel control register field	Programming recommendation
Destination transfer width (peripheral)	32-bit words
Source address increment (memory)	Yes, after each 32-bit transfer.
Destination address increment (peripheral)	Fixed address of CRYP_DIN shall be used (no increment).

Data output using DMA

To enable the DMA for reading data from CRYP processor, set the DOEN bit in the CRYP_DMACR register. When this bit is set, the cryptographic processor initiates a DMA request during the OUTPUT phase each time it requires a word to be read from the CRYP_DOUT register.

Table 264 shows the recommended configuration to transfer data from cryptographic processor to memory through the DMA controller.

Table 264. Cryptographic processor configuration for peripheral-to-memory DMA transfers

DMA channel control register field	Programming recommendation
Transfer size	Message length, multiple of AES cryptographic block size (4 words). *Note: DES/TDES is not supported with DMA.* Depending on the chaining mode selected, extra bits have to be discarded.
Source burst size (peripheral)	CRYP FIFO_size /2 /transfer_width = **4** (FIFO_size = 8x32-bit, transfer_width = 32-bit)
Destination burst size (memory)	If *high latency* memory is used (DDR): CRYP FIFO_size x2 /transfer_width = **16** (FIFO_size = 8x32-bit, transfer_width = 32-bit) If *low latency* memory is used (SRAM): CRYP FIFO_size /2 /transfer_width = **4**
DMA FIFO size	If high latency memory is used (DDR): CRYP FIFO_size x2 = 64 bytes If low latency memory is used (SRAM): CRYP FIFO_size /2 = 16 bytes
Source transfer width (peripheral)	32-bit words
memory transfer width (memory)	32-bit words
Source address increment (peripheral)	Fixed address of CRYP_DOUT shall be used (no increment).
Destination address increment (memory)	Yes, after each 32-bit transfer.

DMA mode

When AES is used, the cryptographic processor manages two DMA transfer requests through `cryp_in_dma` and `cryp_out_dma` internal input/output signals, which are asserted:

- for IN FIFO: every time a block has been read from FIFO by CRYP,
- for OUT FIFO: every time a block has been written into the FIFO by the cryptographic processor.

When DES/TDES is used, the cryptographic processor manages two DMA transfer requests through `cryp_in_dma` and `cryp_out_dma` internal input/output signals, which are asserted:

- for IN FIFO: every time two blocks have been read from FIFO by the cryptographic processor

for OUT FIFO: every time a word has been written into the FIFO by the cryptographic processor (single transfer). Note that a burst transfer is also triggered when two blocks have been written into the FIFO.

Note: DMA cannot be used when DES/TDES algorithms are selected.

All request signals are deasserted if the cryptographic peripheral is disabled or the DMA enable bit is cleared (DIEN bit for the IN FIFO and DOEN bit for the OUT FIFO in the CRYP_DMACR register).

Caution: It is important that DMA controller empties the cryptographic peripheral output FIFO before filling up the CRYP input FIFO. To achieve it, the DMA controller should be configured so that the transfer from the peripheral to the memory has a higher priority than the transfer from the memory to the peripheral.

For more detailed information on DMA operations, refer to *Section 36.4.5: CRYP procedure to perform a cipher operation*.

36.4.20 CRYP error management

No error flags are generated by the cryptographic processor.

36.5 CRYP interrupts

Overview

There are two individual maskable interrupt sources generated by the cryptographic processor to signal the following events:

- Input FIFO empty or not full
- Output FIFO full or not empty

These two sources are combined into a single interrupt signal which is the only interrupt signal from the CRYP processor that drives the Cortex® CPU interrupt controller. You can enable or disable CRYP interrupt sources individually by changing the mask bits in the CRYP_IMSCR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of the individual maskable interrupt sources can be read either from the CRYP_RISR register, for raw interrupt status, or from the CRYP_MISR register for masked interrupt status. The status of the individual source of event flags can be read from the CRYP_SR register.

Table 265 gives a summary of the available features.

Table 265. CRYP interrupt requests

Interrupt acronym	Interrupt event	Event flag in CRYP_xISR[1]	Event flag in CRYP_SR	Enable bit	Interrupt clear method
CRYP	Output FIFO full	OUTRIS (not masked) OUTMIS (masked)	OFFU	OUTIM[2] and CRYPEN	Read one data from output FIFO
	Output FIFO not empty		OFNE		Read all data from output FIFO
	Input FIFO not full	INRIS (not masked) INMIS (masked)	IFNF	INIM[2] and CRYPEN	Write data until input FIFO is full
	Input FIFO empty		IFEM		Write at least one word in input FIFO

1. The flags belong to CRYP_RISR registers (unmasked or raw) or CRYP_MISR registers (masked).
2. The flags belong to CRYP_IMSCR register.

Output FIFO service interrupt - OUTMIS

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is when the interrupt follows the state of the output FIFO not empty flag OFNE).

The output FIFO service interrupt OUTMIS is NOT enabled with the CRYP processor enable bit. Consequently, disabling the CRYP processor does not force the OUTMIS signal low if the output FIFO is not empty.

Input FIFO service interrupt - INMIS

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is enabled with the CRYP enable bit. Consequently, when CRYP is disabled, the INMIS signal is low even if the input FIFO is empty.

36.6 CRYP processing time

The time required to process a block for each mode of operation is summarized below. The block size is 128 bits for AES and 64 bits for DES/TDES.

Table 266. Processing latency for ECB, CBC and CTR

Key size	Operating modes	Chaining algorithm	Clock cycles
64 bits	DES encryption or decryption	ECD, CBC	16
3x64 bits	TDES encryption or decryption		48
128 bits	AES encryption or decryption[1]	ECD, CBC, CTR	14
	AES key preparation	-	12
192 bits	AES encryption or decryption[1]	ECD, CBC, CTR	16
	AES key preparation	-	14
256 bits	AES encryption or decryption[1]	ECD, CBC, CTR	18
	AES key preparation	-	16

1. Excluding key preparation time (ECB and CBC only).

Table 267. Processing time (in clock cycle) for GCM and CCM per 128-bit block

Key size	Operating modes	Chaining algorithm	Initialization phase	Header phase	Payload phase	Tag phase
128 bits	AES encryption or decryption	GCM	24	10	14	14
		CCM	12	14	25	14
192 bits		GCM	28	10	16	16
		CCM	14	16	29	16
256 bits		GCM	32	10	18	18
		CCM	16	18	33	18

36.7 CRYP registers

The cryptographic core is associated with several control and status registers, eight key registers and four initialization vectors registers.

36.7.1 CRYP control register (CRYP_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPBLB[3:0]				ALGOMODE3	Res.	GCM_CCMPH[1:0]	
								rw	rw	rw	rw	rw		rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE[1:0]		DATATYPE[1:0]		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **NPBLB[3:0]:** Number of Padding Bytes in Last Block of payload.

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0000: All bytes are valid (no padding)

0001: Padding for the last LSB byte

...

1111: Padding for the 15 LSB bytes of last block.

Writing NPBLB bits while BUSY = 1 has no effect.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **GCM_CCMPH[1:0]**: GCM or CCM Phase selection

This bitfield has no effect if GCM, GMAC or CCM algorithm is not selected in ALGOMODE field.

00: Initialization phase

01: Header phase

10: Payload phase

11: Final phase

Writing to GCM_CCMPH bits while BUSY = 1 has no effect.

Bit 15 **CRYPEN:** CRYP processor Enable

0: Cryptographic processor peripheral is disabled

1: Cryptographic processor peripheral is enabled

This bit is automatically cleared by hardware when the key preparation process ends (ALGOMODE = 0111) or after GCM/GMAC or CCM Initialization phase.

Bit 14 **FFLUSH:** CRYP FIFO Flush

0: No FIFO flush
1: FIFO flush enabled
When CRYPEN = 0, writing this bit to 1 flushes the IN and OUT FIFOs (that is read and write pointers of the FIFOs are reset). Writing this bit to 0 has no effect. When CRYPEN = 1, writing this bit to 0 or 1 has no effect.
Reading this bit always returns 0.
FFLUSH bit has to be set only when BUSY = 0. If not, the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a non-empty FIFO condition.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:8 **KEYSIZE[1:0]**: Key size selection (AES mode only)

This bitfield defines the bit-length of the key used for the AES cryptographic core. This bitfield is ‘don’t care’ in the DES or TDES modes.
00: 128-bit key length
01: 192-bit key length
10: 256-bit key length
11: Reserved, do not use this value
Writing KEYSIZE bits while BUSY = 1 has no effect.

Bits 7:6 **DATATYPE[1:0]**: Data Type selection

This bitfield defines the format of data written in CRYP_DIN or read from CRYP_DOUT registers. For more details refer to *Section 36.4.16: CRYP data registers and data swapping*).
00: 32-bit data. No swapping for each word. First word pushed into the IN FIFO (or popped off the OUT FIFO) forms bits 1...32 of the data block, the second word forms bits 33...64 etc.
01: 16-bit data, or half-word. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 2 half-words, which are swapped with each other.
10: 8-bit data, or bytes. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 4 bytes, which are swapped with each other.
11: bit data, or bit-string. Each word pushed into the IN FIFO (or popped off the OUT FIFO) is considered as 32 bits (1st bit of the string at position 0), which are swapped with each other.
Writing DATATYPE bits while BUSY = 1 has no effect.

Bits 19, 5:3 **ALGOMODE[3:0]:** Algorithm mode

Below definition includes the bit 19:
0000: TDES-ECB (triple-DES Electronic Codebook).
0001: TDES-CBC (triple-DES Cipher Block Chaining).
0010: DES-ECB (simple DES Electronic Codebook).
0011: DES-CBC (simple DES Cipher Block Chaining).
0100: AES-ECB (AES Electronic Codebook).
0101: AES-CBC (AES Cipher Block Chaining).
0110: AES-CTR (AES Counter mode).
0111: AES key preparation for ECB or CBC decryption.
1000: AES-GCM (Galois Counter mode) and AES-GMAC (Galois Message Authentication Code mode).
1001: AES-CCM (Counter with CBC-MAC).
Writing ALGOMODE bits while BUSY = 1 has no effect.
Others: Reserved, must no be used

Bit 2 **ALGODIR:** Algorithm Direction
0: Encrypt
1: Decrypt

Writing ALGODIR bit while BUSY = 1 has no effect.

Bits 1:0 Reserved, must be kept at reset value.

36.7.2 CRYP status register (CRYP_SR)

Address offset: 0x04

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **BUSY:** Busy bit
0: The CRYP core is not processing any data. The reason is:
– either that the CRYP core is disabled (CRYPEN = 0 in the CRYP_CR register) and the last processing has completed,
– or the CRYP core is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 2 words in the DES, 4 words in the AES).
1: The CRYP core is currently processing a block of data or a key preparation is ongoing (AES ECB or CBC decryption only).

Bit 3 **OFFU:** Output FIFO full flag
0: Output FIFO is not full
1: Output FIFO is full

Bit 2 **OFNE:** Output FIFO not empty flag
0: Output FIFO is empty
1: Output FIFO is not empty

Bit 1 **IFNF:** Input FIFO not full flag
0: Input FIFO is full
1: Input FIFO is not full

Bit 0 **IFEM:** Input FIFO empty flag
0: Input FIFO is not empty
1: Input FIFO is empty

36.7.3 CRYP data input register (CRYP_DIN)

Address offset: 0x08

Reset value: 0x0000 0000

The CRYP_DIN register is the data input register. It is 32-bit wide. It is used to enter into the input FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks (AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to *Section 36.4.16: CRYP data registers and data swapping* for more details.

When CRYP_DIN register is written to the data are pushed into the input FIFO.

- If CRYPEN = 1, when at least four 32-bit words in the AES mode have been pushed into the input FIFO (two words in the DES/TDES mode), and when at least four words are free in the output FIFO (two words in the DES/DTES mode), the CRYP engine starts an encrypting or decrypting process.

When CRYP_DIN register is read:

- If CRYPEN = 0, the FIFO is popped, and then the data present in the Input FIFO are returned, from the oldest one (first reading) to the newest one (last reading). The IFEM flag must be checked before each read operation to make sure that the FIFO is not empty.
- if CRYPEN = 1, an undefined value is returned.

Note: After the CRYP_DIN register has been read once or several times, the FIFO must be flushed by setting the FFLUSH bit prior to processing new data.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATAIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAIN[31:0]:** Data input

On read FIFO is popped (last written value is returned), and its value is returned if CRYPEN = 0. If CRYPEN = 1 DATAIN register returns an undefined value.

On write current register content is pushed inside the FIFO.

36.7.4 CRYP data output register (CRYP_DOUT)

Address offset: 0x0C

Reset value: 0x0000 0000

The CRYP_DOUT register is the data output register. It is read-only and 32-bit wide. It is used to retrieve from the output FIFO up to four 64-bit blocks (TDES) or two 128-bit blocks

(AES) of plaintext (when encrypting) or ciphertext (when decrypting), one 32-bit word at a time.

To fit different data sizes, the data can be swapped after processing by configuring the DATATYPE bits in the CRYP_CR register. Refer to *Section 36.4.16: CRYP data registers and data swapping* for more details.

When CRYP_DOUT register is read, the last data entered into the output FIFO (pointed to by the read pointer) is returned.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DATAOUT[31:0]:** Data output
On read returns output FIFO content (pointed to by read pointer), else returns an undefined value.
On write, no effect.

36.7.5 CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DOEN:** DMA output enable
When this bit is set, DMA requests are automatically generated by the peripheral during the output data phase.
0: Outgoing data transfer from CRYP via DMA is disabled
1: Outgoing data transfer from CRYP via DMA is enabled

Bit 0 **DIEN:** DMA input enable
When this bit is set, DMA requests are automatically generated by the peripheral during the input data phase.
0: Incoming data transfer to CRYP via DMA is disabled
1: Incoming data transfer to CRYP via DMA is enabled

36.7.6 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14

Reset value: 0x0000 0000

The CRYP_IMSCR register is the interrupt mask set or clear register. It is a read/write register. When a read operation is performed, this register gives the current value of the mask applied to the relevant interrupt. Writing 1 to the particular bit sets the mask, thus enabling the interrupt to be read. Writing 0 to this bit clears the corresponding mask. All the bits are cleared to 0 when the peripheral is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTIM:** Output FIFO service interrupt mask
0: Output FIFO service interrupt is masked
1: Output FIFO service interrupt is not masked

Bit 0 **INIM:** Input FIFO service interrupt mask
0: Input FIFO service interrupt is masked
1: Input FIFO service interrupt is not masked

36.7.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18

Reset value: 0x0000 0001

The CRYP_RISR register is the raw interrupt status register. It is a read-only register. When a read operation is performed, this register gives the current raw status of the corresponding interrupt, i.e. the interrupt information without taking CRYP_IMSCR mask into account. Write operations have no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTRIS:** Output FIFO service raw interrupt status
This bit gives the output FIFO interrupt information without taking CRYP_IMSCR corresponding mask into account.
0: Raw interrupt not pending
1: Raw interrupt pending

Bit 0 **INRIS:** Input FIFO service raw interrupt status
This bit gives the input FIFO interrupt information without taking CRYP_IMSCR corresponding mask into account.
0: Raw interrupt not pending
1: Raw interrupt pending

36.7.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C

Reset value: 0x0000 0000

The CRYP_MISR register is the masked interrupt status register. It is a read-only register. When a read operation is performed, this register gives the current masked status of the corresponding interrupt, i.e. the interrupt information taking CRYP_IMSCR mask into account. Write operations have no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTMIS:** Output FIFO service masked interrupt status
This bit gives the output FIFO interrupt information taking into account the corresponding CRYP_IMSCR mask.
0: Output FIFO interrupt not pending
1: Output FIFO interrupt pending

Bit 0 **INMIS:** Input FIFO service masked interrupt status
This bit gives the input FIFO interrupt information taking into account the corresponding CRYP_IMSCR mask.
0: Input FIFO interrupt not pending
1: Input FIFO interrupt pending when CRYPEN = 1

36.7.9 CRYP key register 0L (CRYP_K0LR)

Address offset: 0x20

Reset value: 0x0000 0000

CRYP key registers contain the cryptographic keys.

For more information refer to *Section 36.4.17: CRYP key registers*.

Note: Write accesses to these registers are disregarded when the cryptographic processor is busy (bit BUSY = 1 in the CRYP_SR register) .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[255:224]**: Key bit x (x = 255 to 224)
This write-only bitfield contains the bits [255:224] of the AES encryption or decryption key, depending on the operating mode. This register is not used in DES/TDES mode.

36.7.10 CRYP key register 0R (CRYP_K0RR)

Address offset: 0x24

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[223:192]**: Key bit x (x = 223 to 192)
This write-only bitfield contains the bits [223:192] of the AES encryption or decryption key, depending on the operating mode. This register is not used in DES/TDES mode.

36.7.11 CRYP key register 1L (CRYP_K1LR)

Address offset: 0x28

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[191:160]**: Key bit x (x = 191 to 160)

This write-only bitfield contains the bits [191:160] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [64:33] of the key K1, with parity bits unused.

36.7.12 CRYP key register 1R (CRYP_K1RR)

Address offset: 0x2C

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[159:128]**: Key bit x (x = 159 to 128)

This write-only bitfield contains the bits [159:128] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [32:1] of the key K1, with parity bits unused.

36.7.13 CRYP key register 2L (CRYP_K2LR)

Address offset: 0x30

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[127:96]**: Key bit x (x = 127 to 96)
This write-only bitfield contains the bits [127:96] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [64:33] of the key K2, with parity bits unused.

36.7.14 CRYP key register 2R (CRYP_K2RR)

Address offset: 0x34

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[95:64]**: Key bit x (x = 95 to 64)
This write-only bitfield contains the bits [95:64] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [32:1] of the key K2, with parity bits unused.

36.7.15 CRYP key register 3L (CRYP_K3LR)

Address offset: 0x38

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[63:32]**: Key bit x (x = 63 to 32)

This write-only bitfield contains the bits [63:32] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [64:33] of the key K3, with parity bits unused.

36.7.16 CRYP key register 3R (CRYP_K3RR)

Address offset: 0x3C

Reset value: 0x0000 0000

Refer to *Section 36.7.9: CRYP key register 0L (CRYP_K0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[31:0]**: Key bit x (x = 31 to 0)

This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode. In DES/TDES mode this bitfield contains the bits [32:1] of the key K3, with parity bits unused.

36.7.17 CRYP initialization vector register 0L (CRYP_IV0LR)

Address offset: 0x40

Reset value: 0x0000 0000

The CRYP_IVx(L/R)R registers store the initialization vector or the nonce, depending on the chaining mode selected. The size of the IV data is 64 bits for DES/TDES and 128 bits for AES. For more information refer to *Section 36.4.18: CRYP initialization vector registers*.

Note: Write accesses to these registers are disregarded when the cryptographic processor is busy (BUSY = 1 in the CRYP_SR register).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector bit x (x = 127 to 96)

This bitfield stores the initialization vector bits [127:96] for AES chaining modes other than ECB. In DES/TDES mode it corresponds to IVI bits [63:32].

IV registers are updated by the core after each computation round of the DES/TDES or AES core.

36.7.18 CRYP initialization vector register 0R (CRYP_IV0RR)

Address offset: 0x44

Reset value: 0x0000 0000

Refer to *Section 36.7.17: CRYP initialization vector register 0L (CRYP_IV0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector bit x (x = 95 to 64)

This bitfield stores the initialization vector bits [95:64] for AES chaining modes other than ECB. In DES/TDES mode it corresponds to IVI bits [31:0].

IV registers are updated by the core after each computation round of the DES/TDES or AES core.

36.7.19 CRYP initialization vector register 1L (CRYP_IV1LR)

Address offset: 0x48

Reset value: 0x0000 0000

Refer to *Section 36.7.17: CRYP initialization vector register 0L (CRYP_IV0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector bit x (x = 63 to 32)

This bitfield stores the initialization vector bits [63:32] for AES chaining modes other than ECB. *This register is not used in DES mode*.

IV registers are updated by the core after each computation round of the AES core.

36.7.20 CRYP initialization vector register 1R (CRYP_IV1RR)

Address offset: 0x4C

Reset value: 0x0000 0000

Refer to *Section 36.7.17: CRYP initialization vector register 0L (CRYP_IV0LR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector bit x (x = 31 to 0)

This bitfield stores the initialization vector bits [31:0] for AES chaining modes other than ECB. *This register is not used in DES mode*.

IV registers are updated by the core after each computation round of the AES core.

36.7.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)

Address offset: 0x050 + x* 0x4 (x = 0 to 7)

Reset value: 0x0000 0000

These registers contain the complete internal register states of the CRYP processor when the GCM/GMAC or CCM algorithm is selected. They are useful when a context swap has to be performed because a high-priority task needs the cryptographic processor while it is already in use by another task.

When such an event occurs, the CRYP_CSGCMCCM0..7R and CRYP_CSGCM0..7R (in GCM/GMAC mode) or CRYP_CSGCMCCM0..7R (in CCM mode) registers have to be read and the values retrieved have to be saved in the system memory space. The cryptographic processor can then be used by the preemptive task. Then when the cryptographic computation is complete, the saved context can be read from memory and written back into the corresponding context swap registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSGCMCCMx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CSGCMCCMx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSGCMCCMx[31:0]**: CRYP processor internal register states for GCM, GMAC and CCM modes

Note: This register is not used in DES/TDES or other AES modes than the ones indicated

36.7.22 CRYP context swap GCM registers (CRYP_CSGCMxR)

Address offset: 0x070 + x* 0x4 (x = 0 to 7)

Reset value: 0x0000 0000

Refer to *Section 36.7.21: CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSGCMx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CSGCMx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSGCMx[31:0]**: CRYP processor internal register states for GCM and GMAC modes.

Note: This register is not used in DES/TDES or other AES modes than the ones indicated

36.7.23 CRYP hardware configuration register (CRYP_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0131

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **CFG4[3:0]**: Hardware Generic 4
Reserved, must be kept at reset value.

Bits 11:8 **CFG3[3:0]**: Hardware Generic 3
This field returns the `dma_version` generic value (0x1).

Bits 7:4 **CFG2[3:0]**: Hardware Generic 2
This field returns the `aes_impl` generic value (0x3).

Bits 3:0 **CFG1[3:0]**: Hardware Generic 1
This field returns the `tdes_impl` generic value (0x1).

36.7.24 CRYP version register (CRYP_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER[3:0]				MINVER[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJVER[3:0]**: CRYP processor major version
This field returns the CRYP processor major version.

Bits 3:0 **MINVER[3:0]**: CRYP processor minor version
This field returns the CRYP processor minor version.

36.7.25 CRYP identification (CRYP_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identification code
This field returns the identification code of the CRYP processor.

36.7.26 CRYP size identification register (CRYP_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code

This field returns the size identification code of the CRYP processor as defined below:

SID[31:8] = 0xA3C5DD (fixed code)

SID[7:0] = 0x01 (1-Kbyte address decoding)

36.7.27 CRYP register map

Table 268. CRYP register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRYP_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NPBLB				ALGOMODE[3]	Res.	GCM_CCMPH		CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Res.	Res.
	Reset value									0	0	0	0	0		0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
	Reset value																															0	1

Table 268. CRYP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	CRYP_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
	Reset value																															0	0
0x20	CRYP_K0LR	K[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	K[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
... ...																																	
0x38	CRYP_K3LR	K[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	K[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	CRYP_ CSGCMCCM0R	CSGCMCCM0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	CRYP_ CSGCMCCM1R	CSGCMCCM1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	CRYP_ CSGCMCCM2R	CSGCMCCM2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	CRYP_ CSGCMCCM3R	CSGCMCCM3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	CRYP_ CSGCMCCM4R	CSGCMCCM4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	CRYP_ CSGCMCCM5R	CSGCMCCM5																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	CRYP_ CSGCMCCM6R	CSGCMCCM6																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 268. CRYP register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6C	CRYP_ CSGCMCCM7R	CSGCMCCM7																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	CRYP_CSGCM 0R	CSGCM0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	CRYP_CSGCM 1R	CSGCM1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	CRYP_CSGCM 2R	CSGCM2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	CRYP_CSGCM 3R	CSGCM3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	CRYP_CSGCM 4R	CSGCM4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	CRYP_CSGCM 5R	CSGCM5																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	CRYP_CSGCM 6R	CSGCM6																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	CRYP_CSGCM 7R	CSGCM7																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90 to 0x3EC	Reserved	Reserved																															
0x03F0	CRYP_HWCFG R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value																					0	0	0	1	0	0	1	1	0	0	0	1
0x03F4	CRYP_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJVER [3:0]				MINVER [3:0]			
	Reset value																									0	0	1	0	0	0	1	1
0x03F8	CRYP_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x03FC	CRYP_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

37 Hash processor (HASH)

37.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA2-224, SHA2-256), the MD5 (message-digest algorithm 5) hash algorithm and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of up to (2^{64} – 1) bits. It also computes 128-bit digests for the MD5 algorithm.

37.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 1321, *MD5 Message-Digest Algorithm*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, and MD5
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA2-256) algorithm
  - 66 clock cycles for processing one 512-bit block of data using MD5 algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Word swapping supported: bits, bytes, half-words and 32-bit words
- Automatic padding to complete the input bit string to fit digest minimum block size of 512 bits (16 × 32 bits)
- Single 32-bit input register associated to an internal input FIFO, corresponding to one block size
- AHB slave peripheral, accessible through 32-bit word accesses only (else an AHB error is generated)
- 8 × 32-bit words (H0 to H7) for output message digest
- Automatic data flow control with support of direct memory access (DMA) using one channel.
- Fixed DMA burst transfers of four words supported

- Interruptible message digest computation, on a per-block basis
  – Re-loadable digest registers
  – Hashing computation suspend/resume mechanism, including DMA

37.3 HASH implementation

Table 269. HASH features

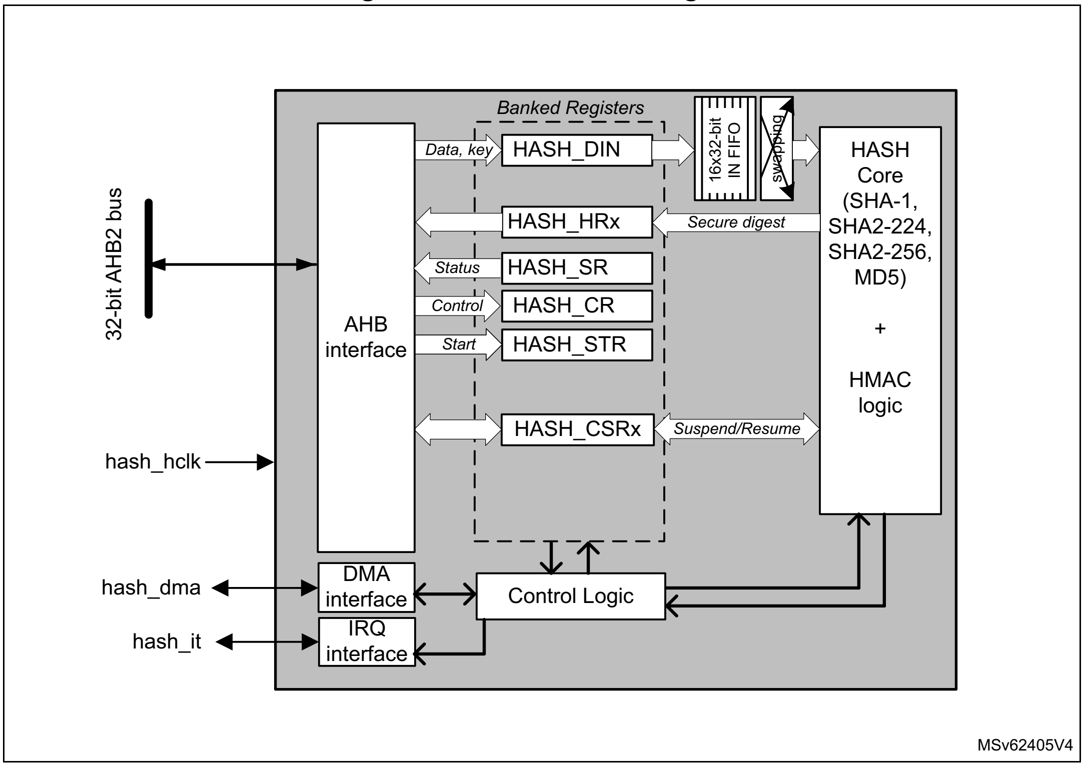
HASH modes/features	HASH1	HASH2
Supported DMA type	Master DMA (MDMA)	Standard DMA
DMA transfer type	Fixed bursts of four words	Single transfers or fixed bursts of four words
DMAA bit (HASH_CR register)	Yes	No

37.4 HASH functional description

37.4.1 HASH block diagram

Figure 309 shows the block diagram of the hash processor.

Figure 309. HASH block diagram



37.4.2 HASH internal signals

Table 270 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 270. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA transfer request/ acknowledge

37.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard and the IETF RFC1321 publication (MD5).

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message of any length below 2^{64} bits is provided on input, the HASH processing core produces respectively a fixed-length output string called a message digest, defined as follows:

- For MD5 digest size is 128-bit
- For SHA-1 digest size is 160-bit
- For SHA2-224 and SHA2-256, the digest size is 224 bits and 256 bits, respectively

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017). Any change to a message in transit, with very high probability, results in a different message digest, and the signature fails to verify.

37.4.4 Message data feeding

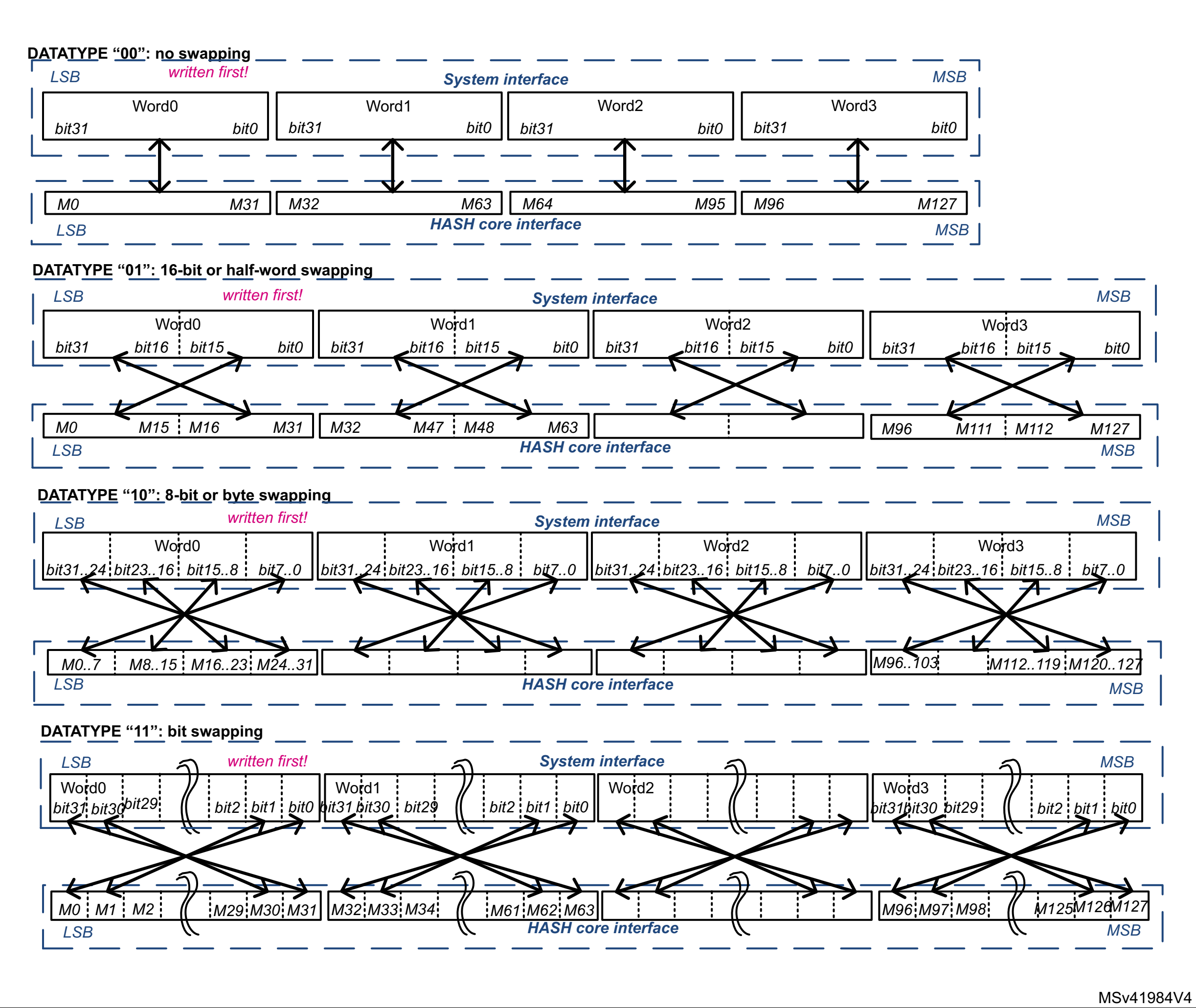
The message (or data file) to be processed by the HASH is considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “`01100001 01100010 01100011`” is represented by a 32-bit word `0x00636261`, and 8-bit words `0x61626300`.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (e.g. byte swapping when data are ASCII text stream) there must be a bit, byte, half-word or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 310* shows how the hash processing core 32-bit data block M0...31 is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR).

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 310. Message data swapping feature



37.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. For MD5, SHA1 and SHA2, the block size is 512 bits.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 37.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO.

In order to perform the hash computation on this data below sequence must be used by the application:

1. Initialize the hash processor using the HASH_CR register:
   a) Select the right algorithm using the ALGO bitfield. If needed program the correct swapping operation on the message input words using DATATYPE bitfield in HASH_CR.
   b) When the HMAC mode is required, set the MODE bit, as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (else keep LKEY cleared). Refer to *Section 37.4.7: HMAC operation* for details.
   c) Update NBLW[4:0] to define the number of valid bits in last word of the message if it is different from 32 bits. NBLW[4:0] information are used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting to 1 the INIT bit in HASH_CR. Also set the bit DMAE to 1 if data are transferred via DMA.

Caution: When programming step 2, it is important to set up before or at the same time the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW[4:0]).

3. Start filling data by writing to HASH_DIN register, unless data are automatically: transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   a) When data are filled by software:
   – Partial digest computation are triggered each time the application writes the first word of the next block. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
   – The final digest computation is triggered when the last block is entered and the software writes the DCAL bit to 1. If the message length is not an exact multiple of the block size, the NBLW[4:0] bitfield in HASH_STR register must be written prior to writing DCAL bit (see *Section 37.4.6* for details).
   b) When data are filled by DMA as a single DMA transfer (MDMAT bit = 0):
   – Partial digest computations are triggered automatically each time the FIFO is full.
   The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register (DCAL bit is set to 1 by hardware). If the message length is not an exact multiple of the block size, the NBLW[4:0] field

in HASH_STR register must be written prior to enabling the DMA (see *Section 37.4.6* for details).

c) When data are filled by DMA using multiple DMA transfers (MDMAT bit = 1) and HASH_HWCFGR register equals 0x0:

– Partial digest computations are triggered as for single DMA transfers. However the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set to 1 by hardware). It allows the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must set MDMAT bit to 0 before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers (see description before).

d) When data are is filled by MDMA (HASH_HWCFGR register is equal to 0x1):

– The partial digest computation is triggered automatically each time the FIFO is full.

– The final digest computation is not triggered automatically when the last block has been transferred to the HASH_DIN register. When the second last block transfer of 4 words is performed by the DMA, the software must set the DMA Abort bit (DMAA) to 1 before completing the last transfer of 4 words or less into HASH_DIN (the actual number of words depends on the message total length). Once the last word has been entered, the software sets DCAL bit to 1 to start the final digest computation process. If the message length is not an exact multiple of 512 bits, the NBLW[4:0] field in HASH_STR register must be written prior to writing DCAL bit (see *Section 37.4.6* for details).

4. Once the digest computation is complete (DCIS = 1), the resulting digest can be read from the output registers as described in *Table 271*.

Table 271. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
MD5	HASH_H0 to HASH_H3	HASH_H0[31]	128
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256

For more information about HMAC detailed instructions, refer to *Section 37.4.7: HMAC operation*.

37.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW[4:0], has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed padding sequences with DMA enabled or disabled are described in *Section 37.4.5: Message digest computing*.

Padding example

As specified by Federal Information Processing Standards PUB 180-4, the message padding consists in appending a “1” followed by *k* “0”s, itself followed by a 64-bit integer that is equal to the length *L* in bits of the message. These three padding operations generate a padded message of length *L* + 1 + *k* + 64, which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH_DIN register at the bit position defined by the NBLW[4:0] bitfield, and the remaining upper bits are cleared (“0”s).

Example from FIPS PUB180-4

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24:

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW[4:0] has to be loaded with the value 24: a **“1”** is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

The message length value, L, in two-word format (that is 00000000 00000018) is appended. Hence the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered by following the below sequence:

1. `0xUU636261` is written to the HASH_DIN register (where ‘U’ means don’t care).
2. `0x18` is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits).
3. `0x10` is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.

4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0...4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

37.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
                  Hash((Key | pad) XOR ipad | message))
```

where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the size of the underlying hash function data block (n = 64 for 512-bit blocks).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n,* the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software writes the INIT bit to 1 with the MODE bit at 1 and the ALGO bits set to the value corresponding to the desired algorithm. The LKEY bit must also be set to 1 if the key being used is longer than 64 bytes. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading, that is writing the key data into HASH_DIN register then completing the transfer by writing DCAL bit to 1 and the correct NBLW[4:0] to HASH_STR register.

Note: Endianness details can be found in Section 37.4.4: Message data feeding.

3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software writes DCAL bit to 1 in HASH_STR register, the NBLW[4:0] bitfield must be written at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 37.4.4: Message data feeding.*
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register

then completing the transfer by writing DCAL bit to 1 and the correct NBLW[4:0] to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set to 1.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 37.4.11: HASH processing time.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST.

Let us assume that the original message is the ASCII binary-coded form of “`Sample message for keylen = blocklen`”, of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is no padding), specified by NIST. As key length = 64, LKEY bit is set to 0 in HASH_CR register

```
00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
30313233 34353637 38393A3B 3C3D3E3F
```

2. **Message** input (length = 34, that is padding required). HASH_STR must be set to `0x20` to start message padding and inner hash computation (see ‘U’ as don’t care)

```
53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
6E3D626C 6F636B6C 656EUUUU
```

3. **Outer hash key** input (length = 64, that is no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers (x = 0 to 4), as shown below:

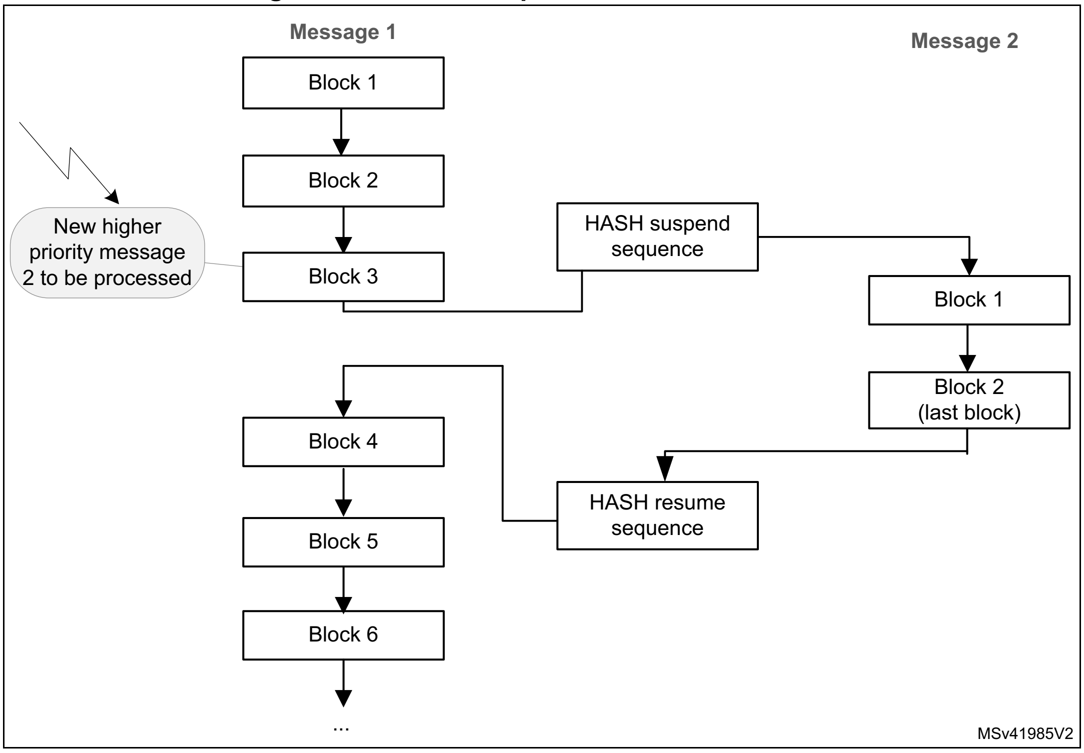
```
HASH_HR0 = 0x5FD596EE
HASH_HR1 = 0x78D5553C
HASH_HR2 = 0x8FF4E72D
HASH_HR3 = 0x266DFD19
HASH_HR4 = 0x2366DA29
```

37.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 311*.

Figure 311. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing 16 words 32-bit (plus one if it is the first block):

1. In Polling mode, wait for BUSY = 0, then poll if the DINIS status bit is set to 1.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR and HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Note: To optimize the resume process when NBW[3:0] = 0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing is described below.

To suspend the processing of a message using DMA, proceed as follows:

1. In Polling mode, wait for BUSY = 0. If DCIS is set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 2.
2. In Polling mode, wait for BUSY = 1.
3. Disable the DMA channel. Then clear DMAE bit in HASH_CR register.
4. In Polling mode, wait for BUSY = 0. If DCIS is set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR, HASH_CR, and HASH_CSR0 to HASH_CSR37 registers. HASH_CSR38 to HASH_CSR53 registers must also be saved if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again. Do not forget to take into account the words that have been already pushed into the FIFO if NBW[3:0] is higher than 0x0.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

Note: To optimize the resume process when NBW[3:0] = 0x0, HASH_CSR22 to HASH_CSR37 registers do not need to be saved then restored as the FIFO is empty.

37.4.9 HASH DMA interface

The HASH only supports fixed DMA burst transfers of four words (if indicated in *Section 37.3: HASH implementation*).

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

When the HASH_HWCFGR register is equal to 0x1, only burst transfers of four words are supported.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information refer to *Section 37.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW[4:0] = Len(Message)% 32` where “`x%32`” gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

37.4.10 HASH error management

No error flags are generated by the hash processor.

37.4.11 HASH processing time

Table 272 summarizes the time required to process an intermediate block for each mode of operation.

Table 272. Processing time (in clock cycle)

Mode of operation	FIFO load(1)	Computation phase	Total
MD5	16	50	**66**
SHA-1	16	66	**82**
SHA2-224	16	50	**66**
SHA2-256			

1. Add the time required to load the block into the processor.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- ~**2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- ~**2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

37.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the NVIC (nested vectored interrupt controller). Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit to 1 enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 273* gives a summary of the available features.

Table 273. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

37.6 HASH registers

The HASH core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

37.6.1 HASH control register (HASH_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO1	Res.	LKEY
													rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DMAA	MDMAT	DINNE	NBW[3:0]				ALGO0	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	rw	rw	r	r	r	r	r	rw	rw	rw	rw	rw	rw		

Bits 31:19 Reserved, must be kept at reset value.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LKEY:** Long key selection

This bit selects between short key (≤ 64 bytes) or long key (> 64 bytes) in HMAC mode.

0: the HMAC key is shorter or equal to 64 bytes. The actual key value written to HASH_DIN is used during the HMAC computation.

1: the HMAC key is longer than 64 bytes. The hash of the key is used instead of the real key during the HMAC computation.

This selection is only taken into account when the INIT and MODE bits are both set. Changing this bit during a computation has no effect.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DMAA:** DMA Abort

This bit is used to abort the usage of DMA to compute a digest. When DMAA is set DMA requests are no more issued towards the DMA, with DMAE in control register and DMAS in status register both cleared.

If DMAA is read, will always return 0. Writing 0 has no effect.

Note: The availability of DMAA depends on peripheral implementation. Refer to Section 37.3: HASH implementation.

Bit 13 **MDMAT:** Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE:** DIN not empty

This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is written to 1.

0: No data are present in the data input buffer

1: The input buffer contains at least one word of data

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

This bitfield reflects the number of words in the message that have already been pushed into the IN FIFO. NBW is incremented by one when a write access to the HASH_DIN register is performed (except if DINNE = 0 and the DMA is not used, see below description). NBW goes to zero when the INIT bit is written to 1.

This bitfield is read-only.

If the DMA is not used

0000: if DINNE = 0, no word has been pushed into the DIN buffer (both HASH_DIN register and IN FIFO are empty), otherwise one word has been pushed into the DIN buffer (HASH_DIN register contains one word and IN FIFO is empty)

0001: two words have been pushed into the DIN buffer (that is HASH_DIN register and the IN FIFO contain one word each)

...

1111: 16 words have been pushed into the DIN buffer.

If the DMA is used

NBW contains the exact number of words that have been pushed into the IN FIFO by the DMA.

Bits 18, 7 **ALGO[1:0]**: Algorithm selection

These bits select the hash algorithm.

00: SHA-1

01: MD5

10: SHA2-224

11: SHA2-256

This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect.

Bit 6 **MODE:** Mode selection

This bit selects the HASH or HMAC mode for the selected algorithm:

0: Hash mode selected

1: HMAC mode selected. LKEY must be set if the key being used is longer than 64 bytes.

This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

Defines the format of the data entered into the HASH_DIN register:

00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.

01: 16-bit data, or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.

10: 8-bit data, or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.

11: bit data, or bit-string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the HASH core is ready to receive data.
After this bit is set it is cleared by hardware while the last data of the message is written into the hash processor.
Setting this bit to 0 while a DMA transfer is ongoing is not aborting this current transfer. Instead, the DMA interface of the IP remains internally enabled until the transfer is completed or INIT is written to 1.
Setting INIT bit to 1 does not clear DMAE bit.

Bit 2 **INIT:** Initialize message digest calculation
Writing this bit to 1 resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.
Writing this bit to 0 has no effect. Reading this bit always return 0.

Bits 1:0 Reserved, must be kept at reset value.

37.6.2 HASH data input register (HASH_DIN)

Address offset: 0x04

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks. When the HASH_DIN register is programmed, the value presented on the AHB databus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB databus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing new data into the HASH_DIN register (the first word of the next block) if the DMA is not used (intermediate digest calculation),
- automatically if the DMA is used.

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by writing the DCAL bit to 1 in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is set to 0 and DMAA is not set to 1.

Reading the HASH_DIN register returns the last word written to this location (zero after reset).

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATAIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the IN FIFO, and the register takes the new value presented on the AHB databus.

Reading this register returns the current register content.

37.6.3 HASH start register (HASH_STR)

Address offset: 0x08

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register).
- It is used to start the processing of the last block in the message by writing the DCAL bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							rw				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

Writing this bit to 1 starts the message padding, using the previously written value of NBLW[4:0], and starts the calculation of the final message digest with all data words written to the input FIFO since the INIT bit was last written to 1.

Reading this bit returns 0.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written in HASH_DIN register, the hash processor takes only the valid bits specified as below, after internal data swapping:

0x00: All 32 bits of the last data written are valid message bits that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

The above mechanism is valid only if DCAL = 0. If NBLW[4:0] bitfield is written while DCAL is set to 1, the NBLW[4:0] bitfield remains unchanged. In other words it is not possible to configure NBLW[4:0] and set DCAL at the same time.

Reading NBLW[4:0] bitfield returns the last value written to NBLW[4:0].

37.6.4 HASH digest registers

These registers contain the message digest result named as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result
- HASH_HR0, HASH_HR1, HASH_HR2 and HASH_HR3 registers return A, B, C and D (respectively), as defined by MD5.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256 digest result.

In all cases, the digest most significant bit is stored in HASH_H0[31] and unused HASH_HRx registers are read as zeros.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation is stalled until the hash calculation has completed.

Note: When starting a digest computation for a new message (by writing the INIT bit to 1), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses.

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x0C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]**: Hash data x
Refer to *Section 37.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]**: Hash data x
Refer to *Section 37.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]**: Hash data x
Refer to *Section 37.6.4: HASH digest registers* introduction.

37.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

37.6.6 HASH status register (HASH_SR)

Address offset: 0x24

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by writing the INIT bit to 1 in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)
1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.
When DINIS=0, HASH_CSRx registers reads as zero.

37.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task, and when the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + x * 0x4, (x = 0 to 53)

Reset value: 0x0000 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (HASH_CSR1 to 53)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSx[31:0]**: Context swap x

Refer to *Section 37.6.7: HASH context swap registers* introduction.

37.6.8 HASH hardware configuration register (HASH_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG1[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CFG1[3:0]**: Hardware generic 1

Reflects the supported type of DMA.

37.6.9 HASH version register (HASH_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: HASH major revision

Bits 3:0 **MINREV[3:0]**: HASH minor revision

37.6.10 HASH identification register (HASH_IPIDR)

Address offset: 0x3F8

Reset value: 0x0017 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: HASH identifier

37.6.11 HASH size identification register (HASH_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification code
This bitfield returns the size identification code of the RNG peripheral as defined below:
Bits[31:8] = 0xA3C5DD (fixed code)
Bits[7:0] = 0x01 (1 Kbyte address decoding)

37.6.12 HASH register map

Table 274. HASH register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[1]	Res.	LKEY	Res.	DMAA	MDMAT	DINNE	NBW[3:0]				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Res.	Res.
	Reset value														0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		
0x04	HASH_DIN	DATAIN[31:16]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 274. HASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x0C	HASH_HRA0	H0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	HASH_HRA1	H1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	HASH_HRA2	H2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	HASH_HRA3	H3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	HASH_HRA4	H4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x24	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1
0x28- 0xF4	Reserved	Reserved																															
0x0F8	HASH_CSR0	CS0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0F8 + 0x4 * x, (x=1 to 53) Last address: 0x1CC	HASH_CSRx	CSx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1D0- 0x30C	Reserved	Reserved																															
0x310	HASH_HR0	H0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	HASH_HR1	H1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x318	HASH_HR2	H2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x31C	HASH_HR3	H3[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x320	HASH_HR4	H4[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x324	HASH_HR5	H5[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 274. HASH register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x328	HASH_HR6	H6[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x32C	HASH_HR7	H7[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03F0	HASH_ HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				Res.				Res.				CFG1			
	Reset value	0x0000 0001																															
0x03F4	HASH_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV				MINREV			
	Reset value	0x0000 0023																															
0x03F8	HASH_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
0x03FC	HASH_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

38 Advanced-control timers (TIM1/TIM8)

38.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

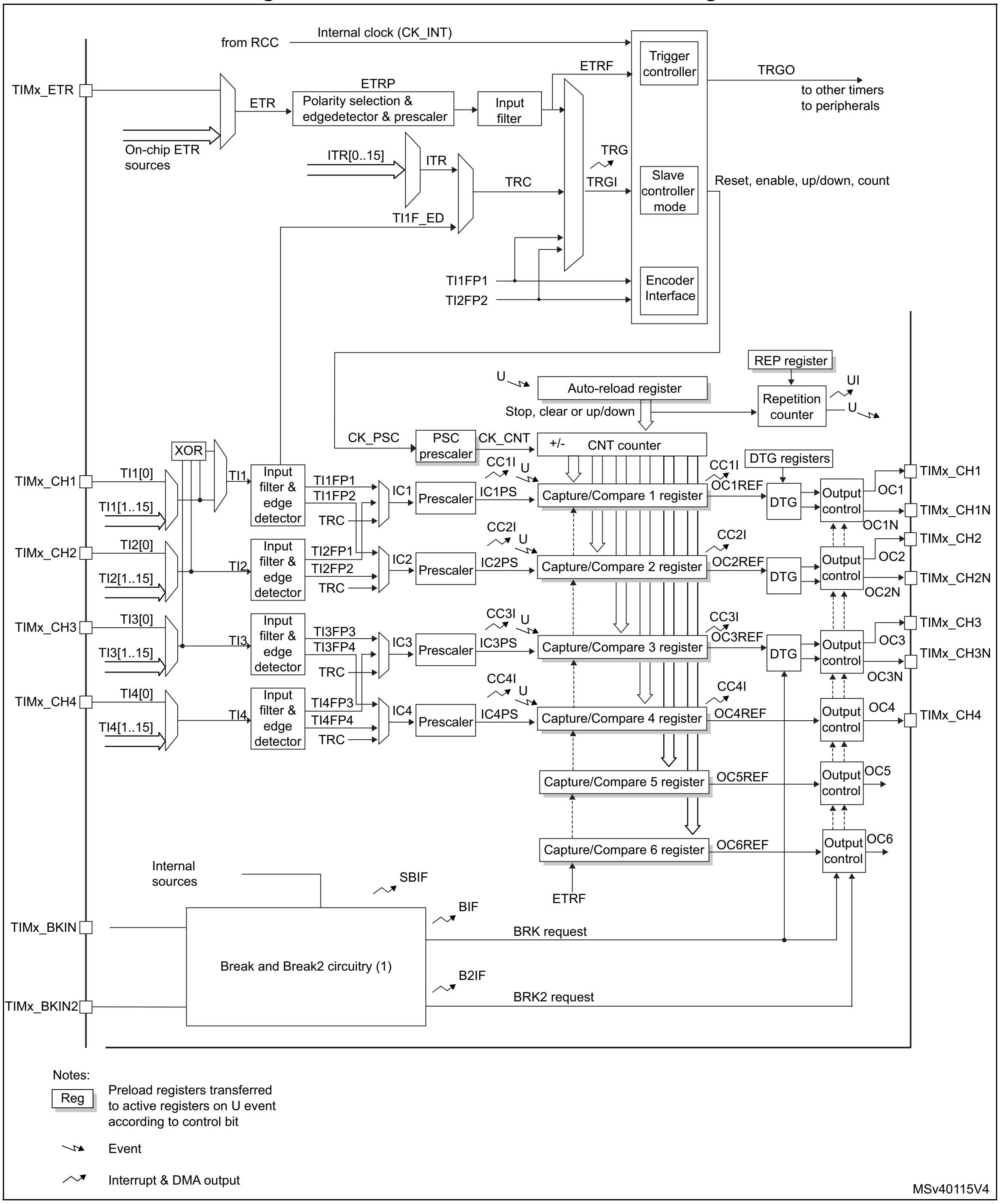
The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 38.3.26: Timer synchronization*.

38.2 TIM1/TIM8 main features

TIM1/TIM8 timer features include:

- 16-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 6 independent channels for:
  – Input Capture (but channels 5 and 6)
  – Output Compare
  – PWM generation (Edge and Center-aligned Mode)
  – One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  – Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  – Trigger event (counter start, stop, initialization or count by internal/external trigger)
  – Input capture
  – Output compare
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 312. Advanced-control timer block diagram



1. See *Figure 355: Break and Break2 circuitry overview* for details

38.3 TIM1/TIM8 functional description

38.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 313 and *Figure 314* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 313. Counter timing diagram with prescaler division change from 1 to 2

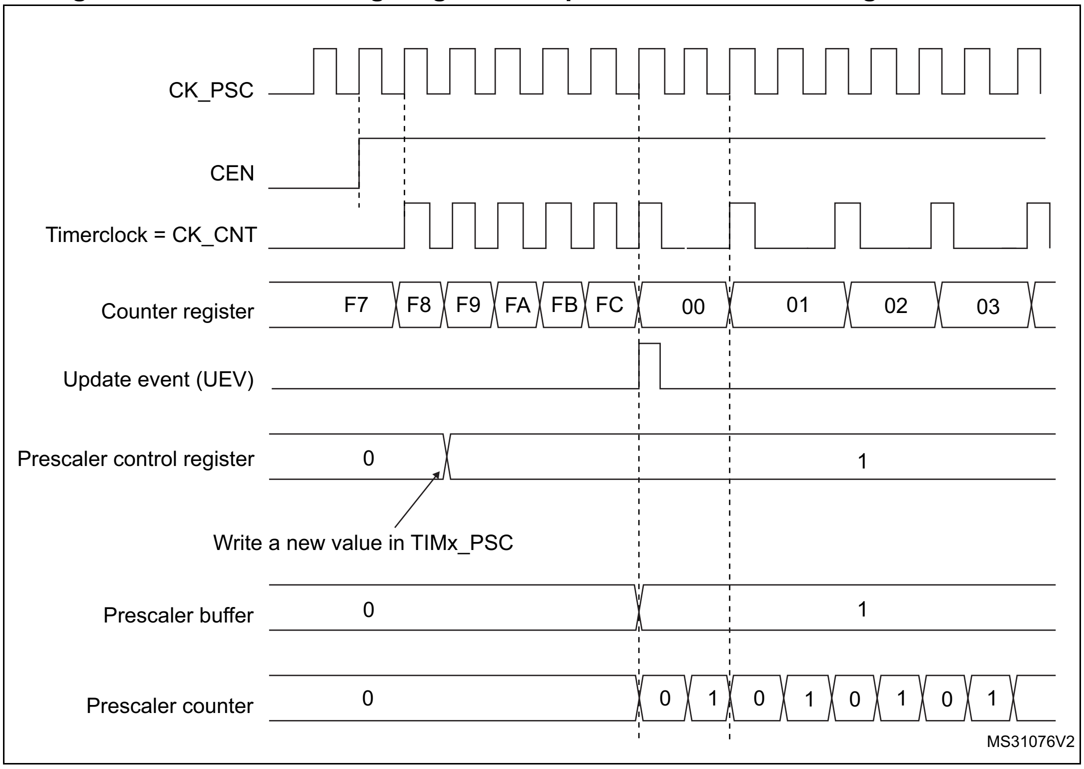
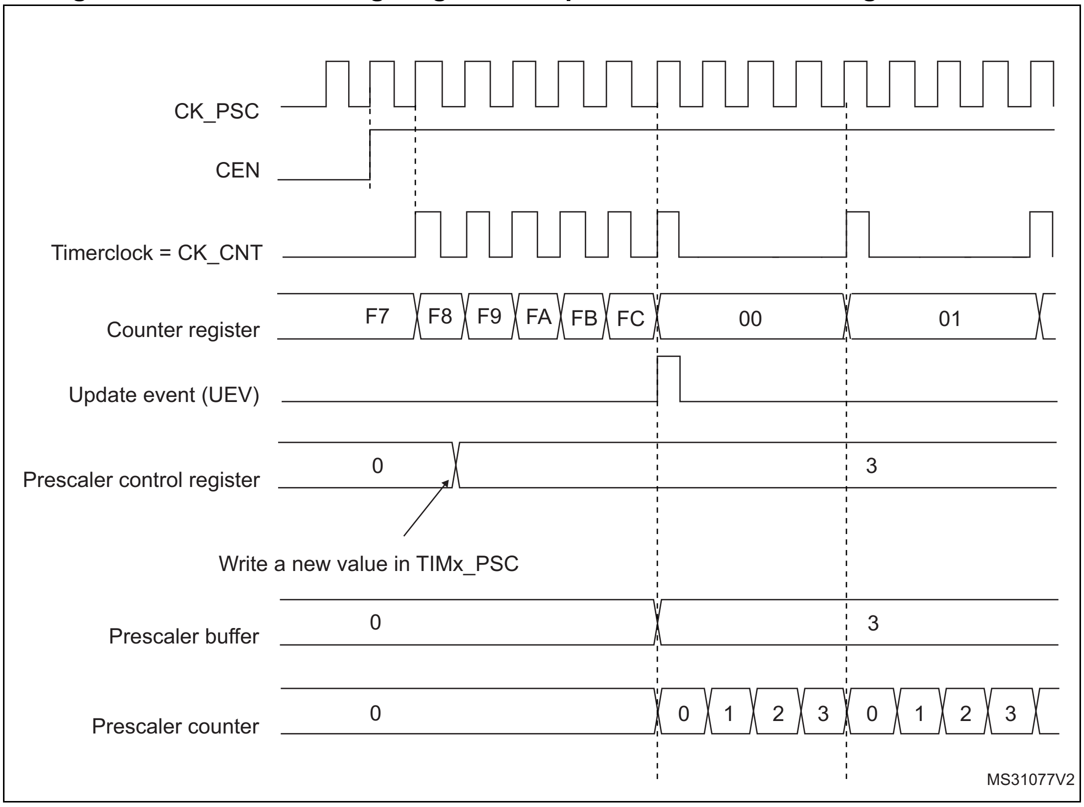


Figure 314. Counter timing diagram with prescaler division change from 1 to 4



38.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 315. Counter timing diagram, internal clock divided by 1

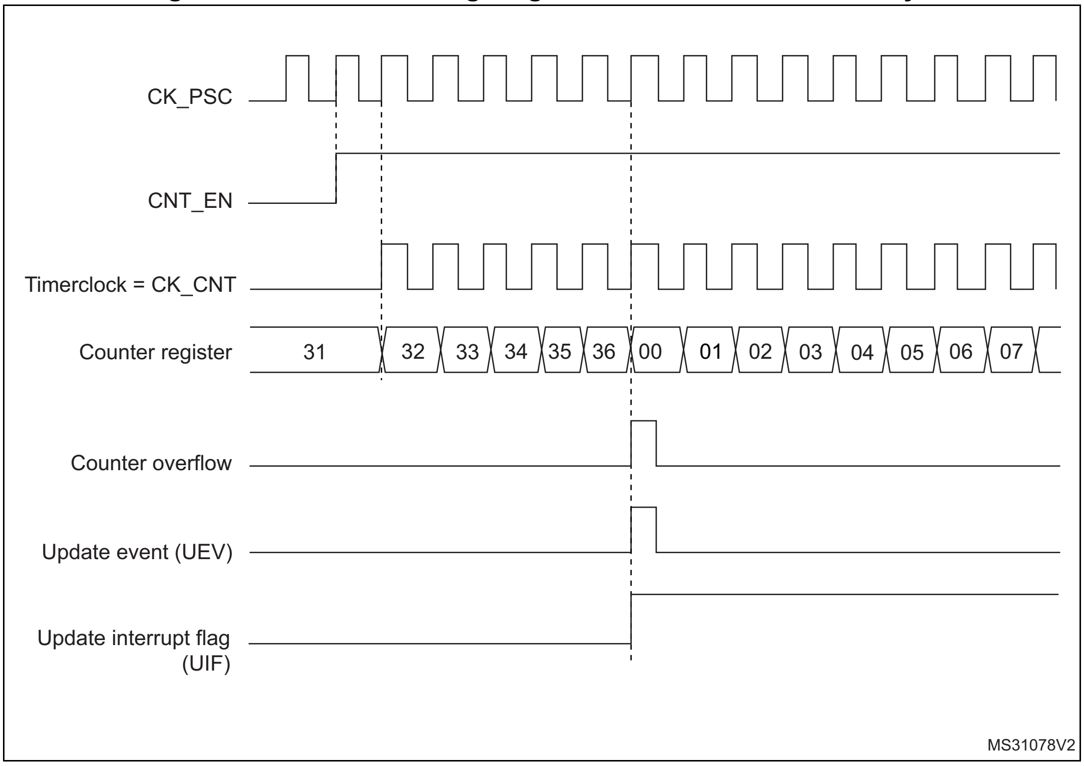


Figure 316. Counter timing diagram, internal clock divided by 2

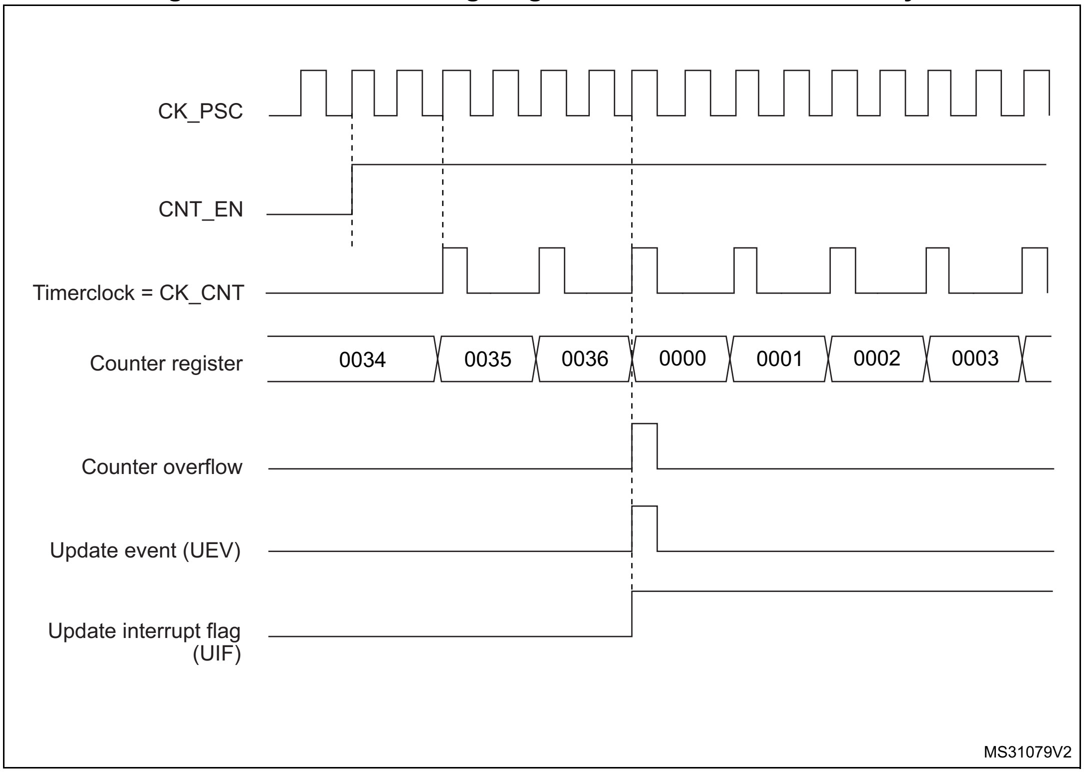


Figure 317. Counter timing diagram, internal clock divided by 4

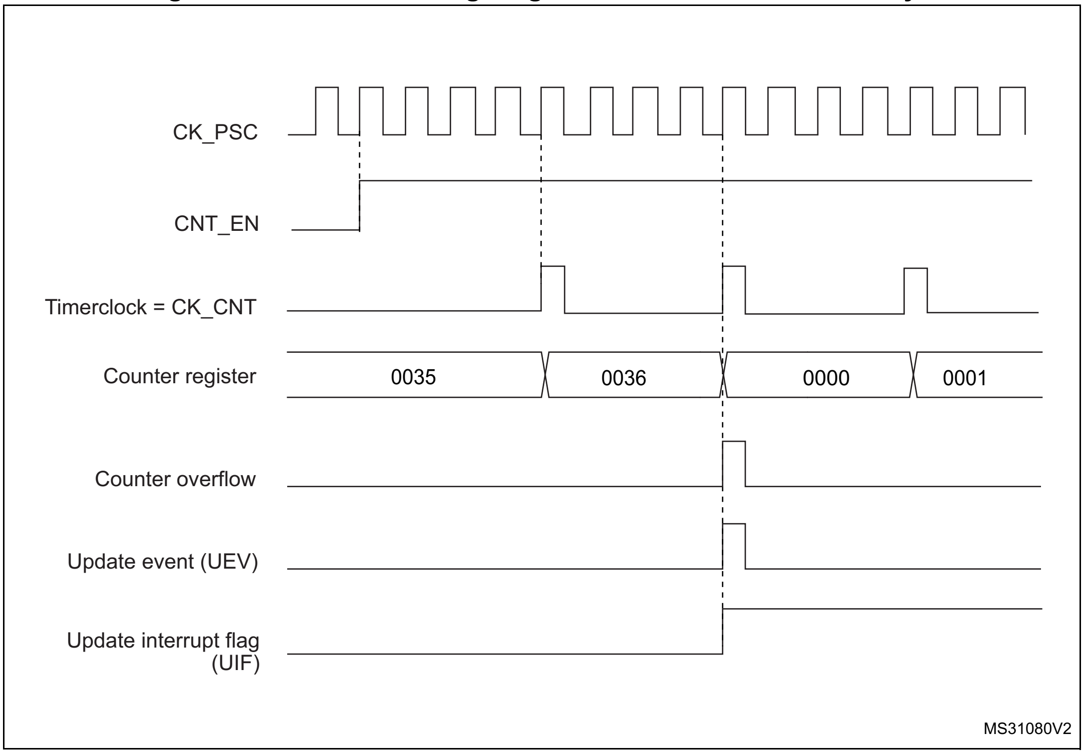


Figure 318. Counter timing diagram, internal clock divided by N

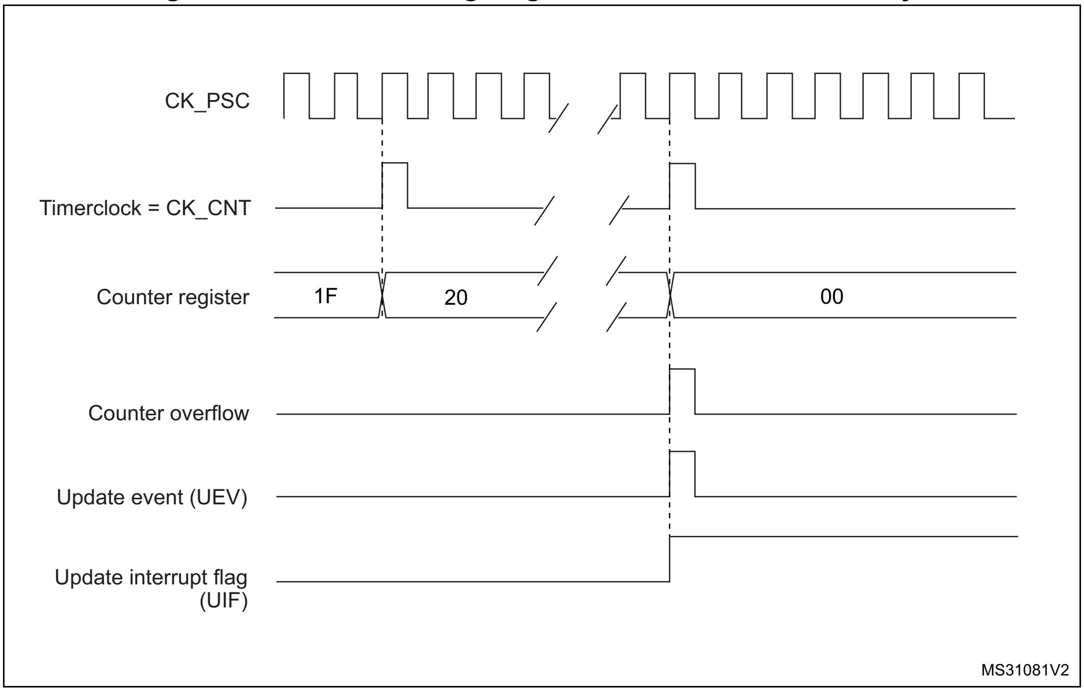


Figure 319. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

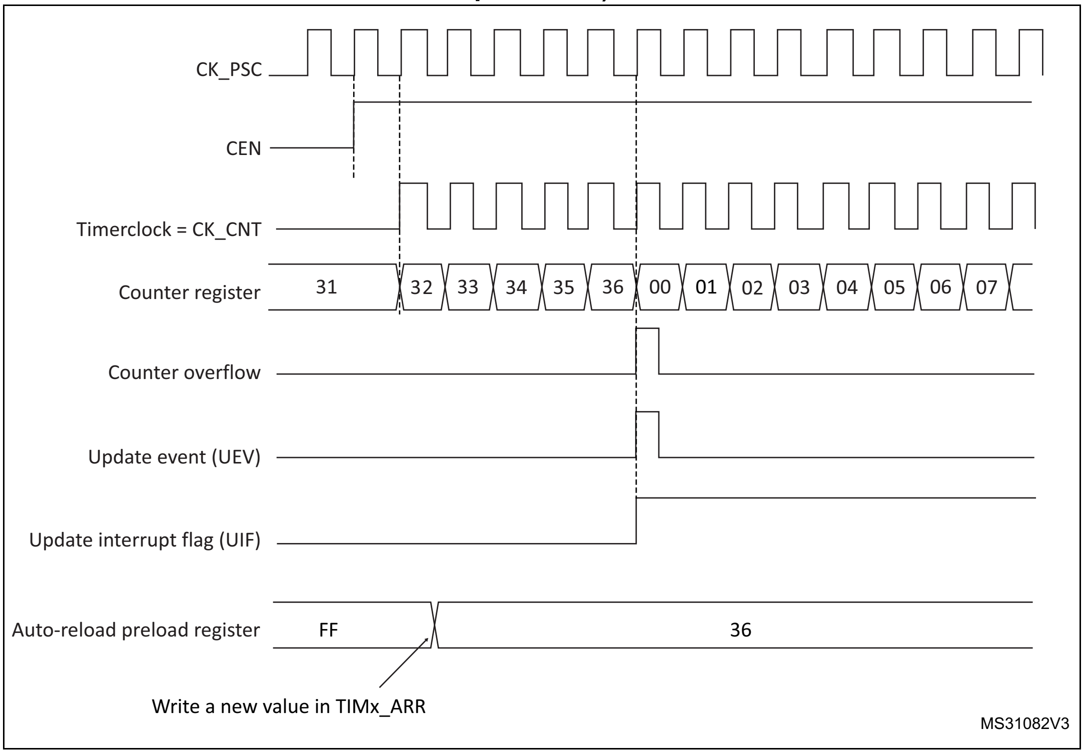
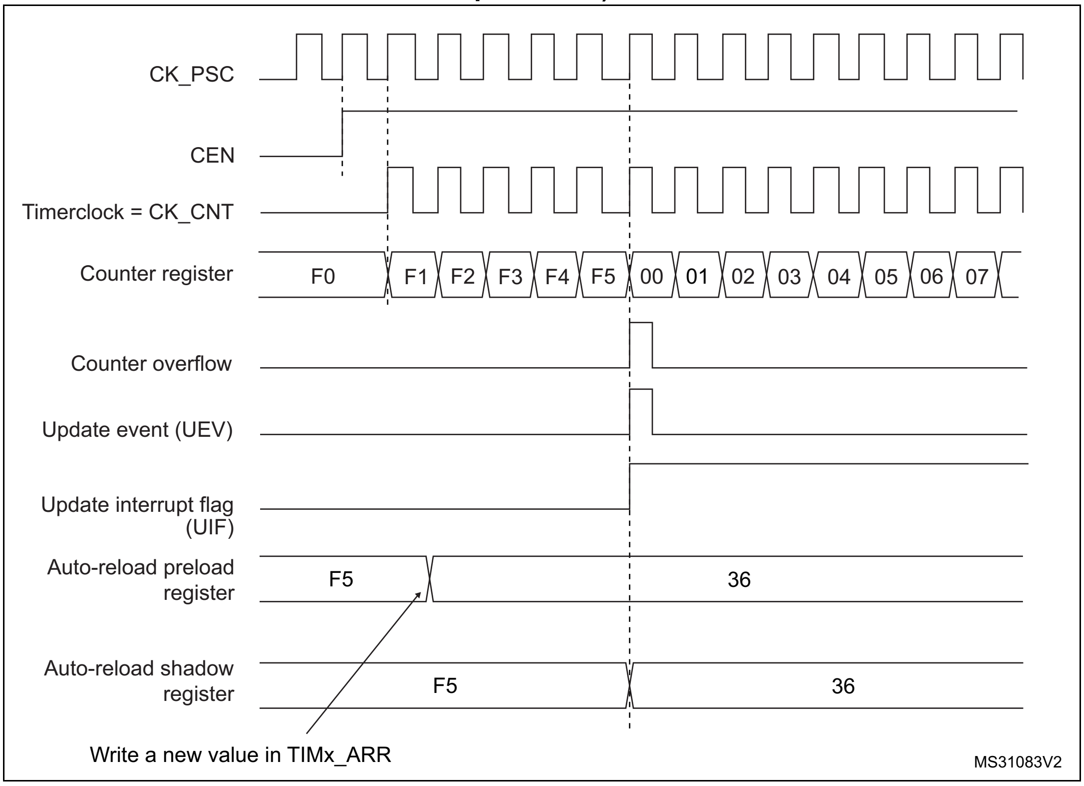


Figure 320. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 321. Counter timing diagram, internal clock divided by 1

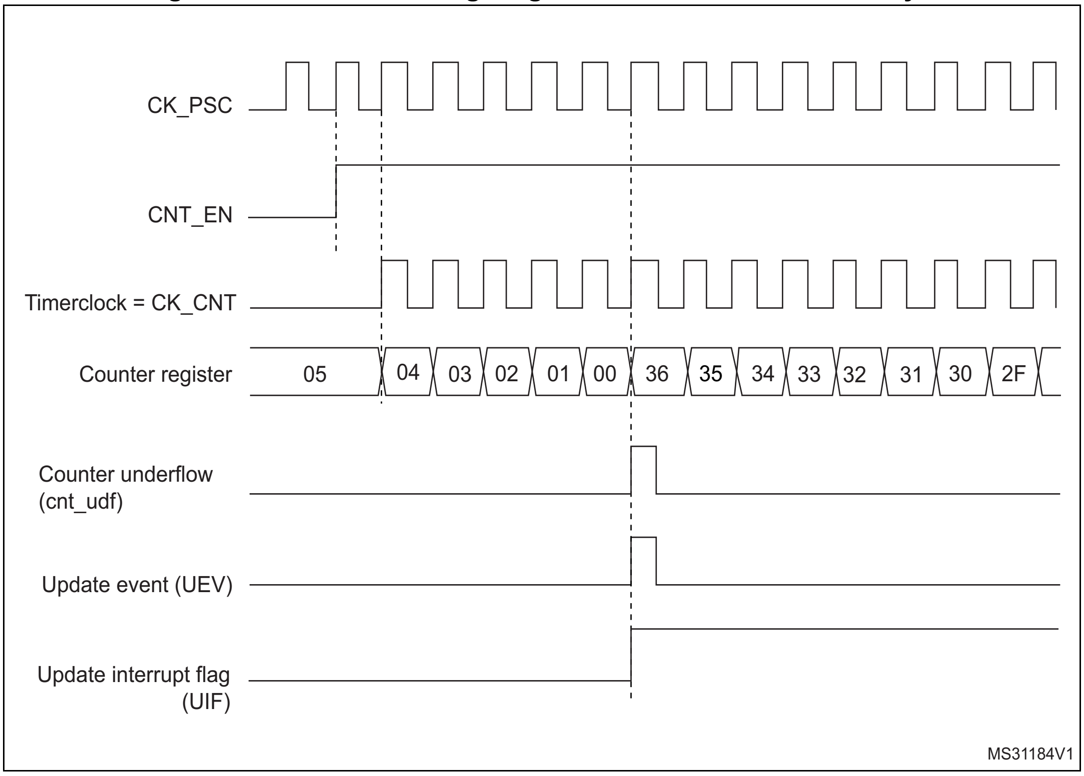


Figure 322. Counter timing diagram, internal clock divided by 2

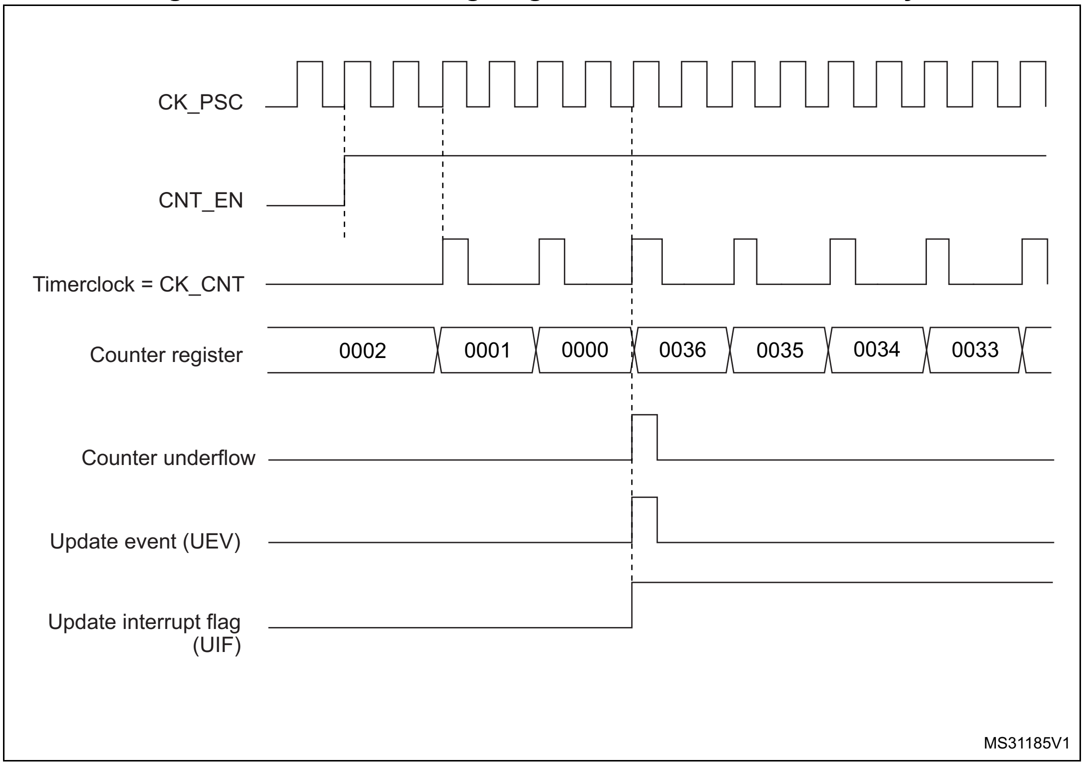


Figure 323. Counter timing diagram, internal clock divided by 4

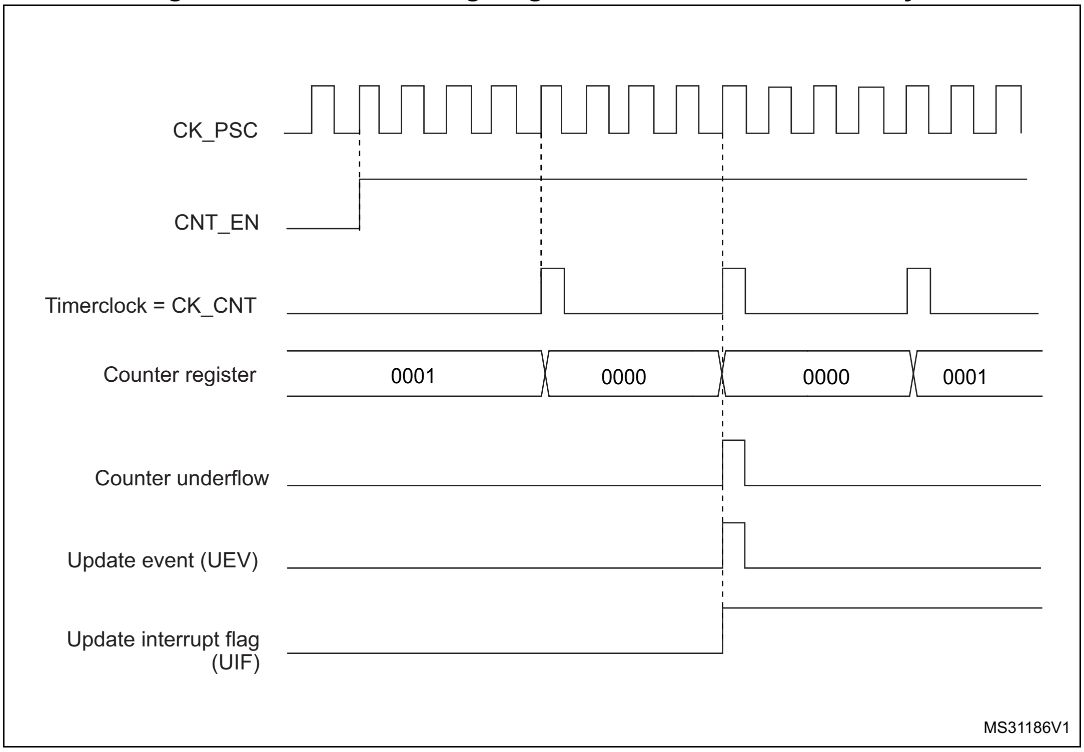


Figure 324. Counter timing diagram, internal clock divided by N

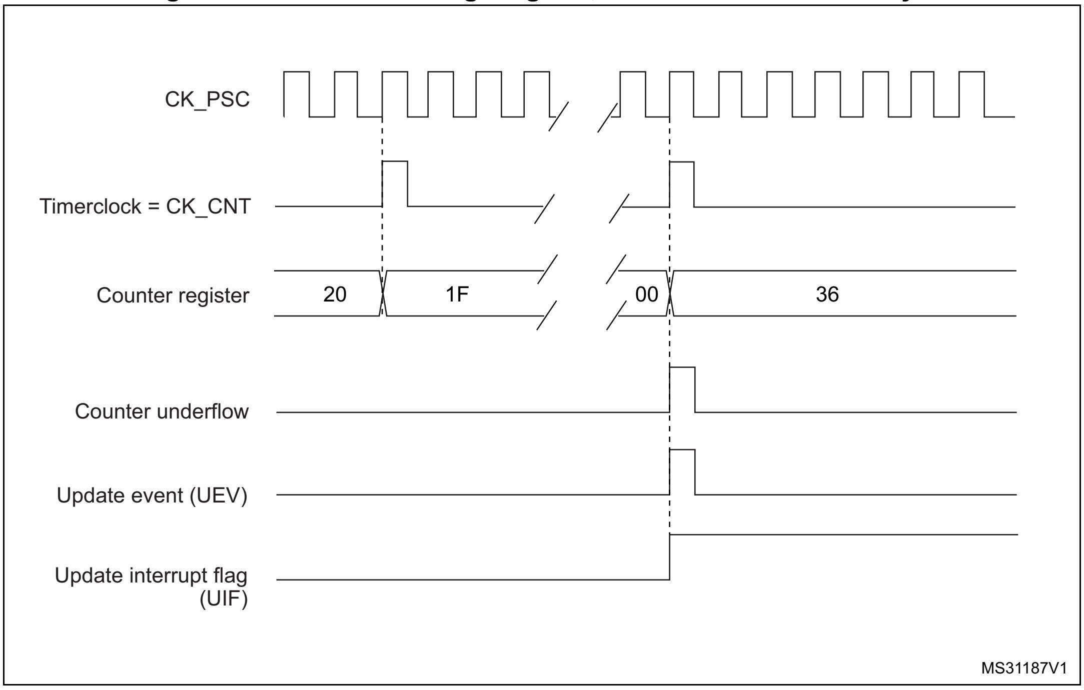
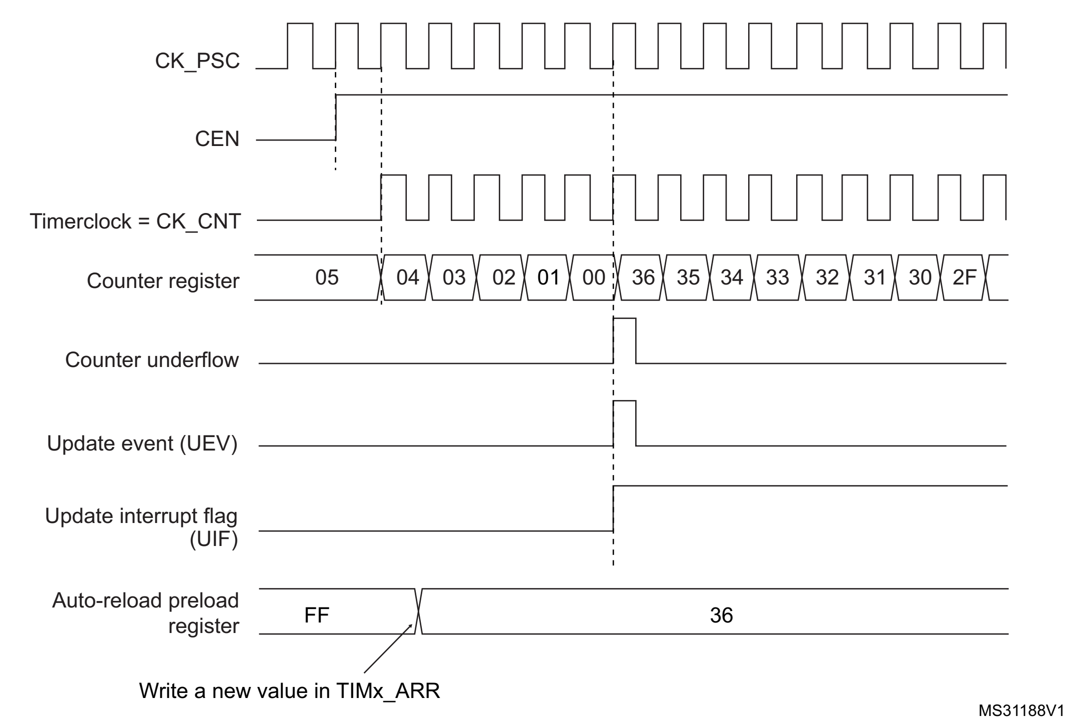


Figure 325. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or

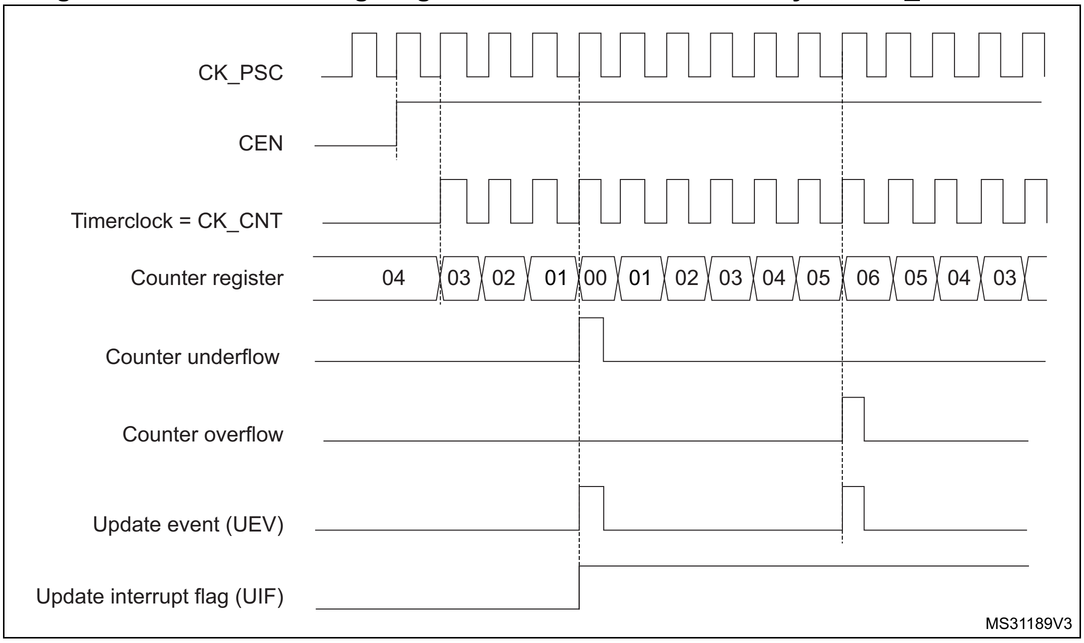
DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 326. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 38.4: TIM1/TIM8 registers*).

Figure 327. Counter timing diagram, internal clock divided by 2

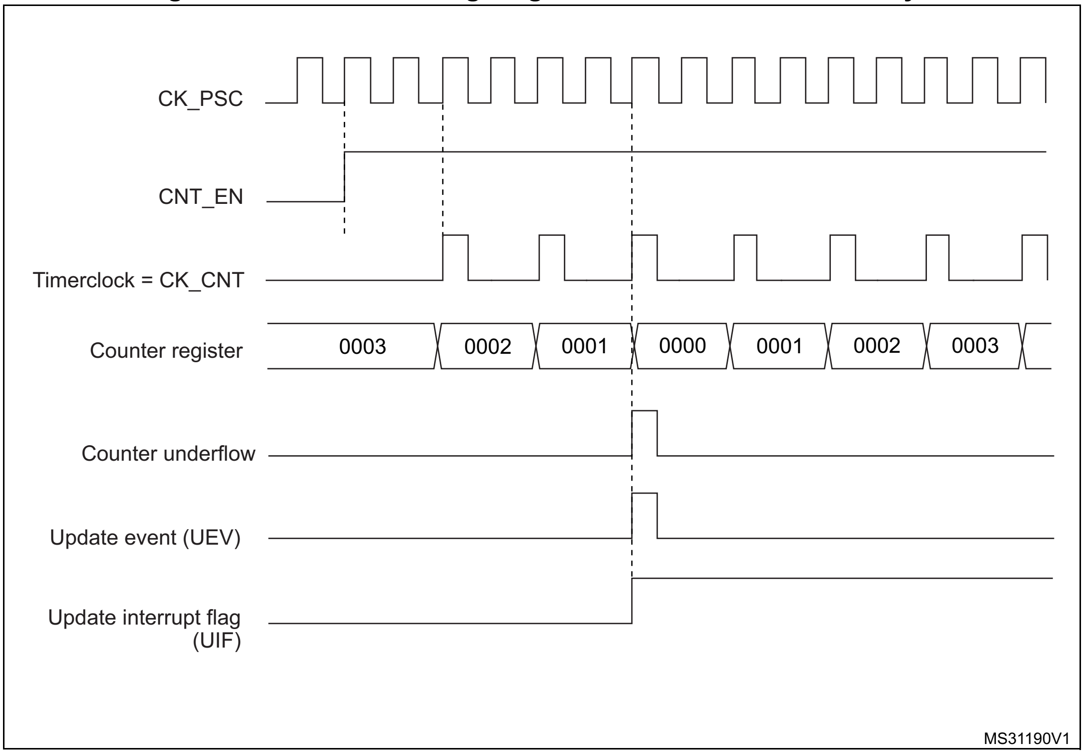


Figure 328. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36

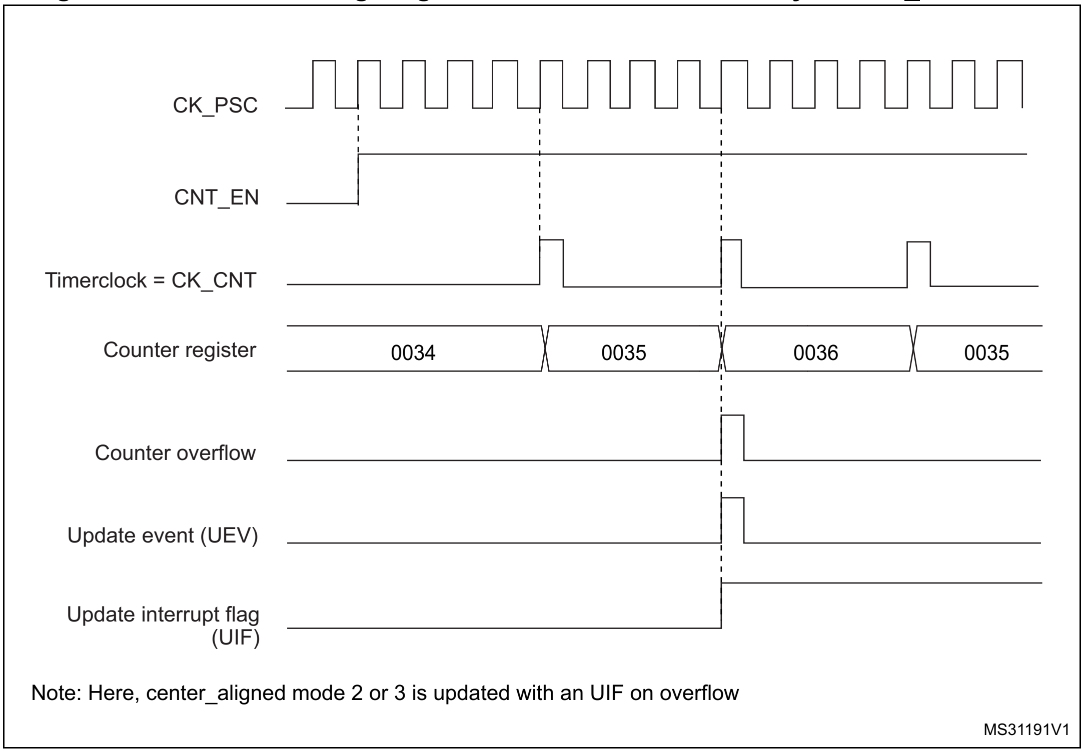


Figure 329. Counter timing diagram, internal clock divided by N

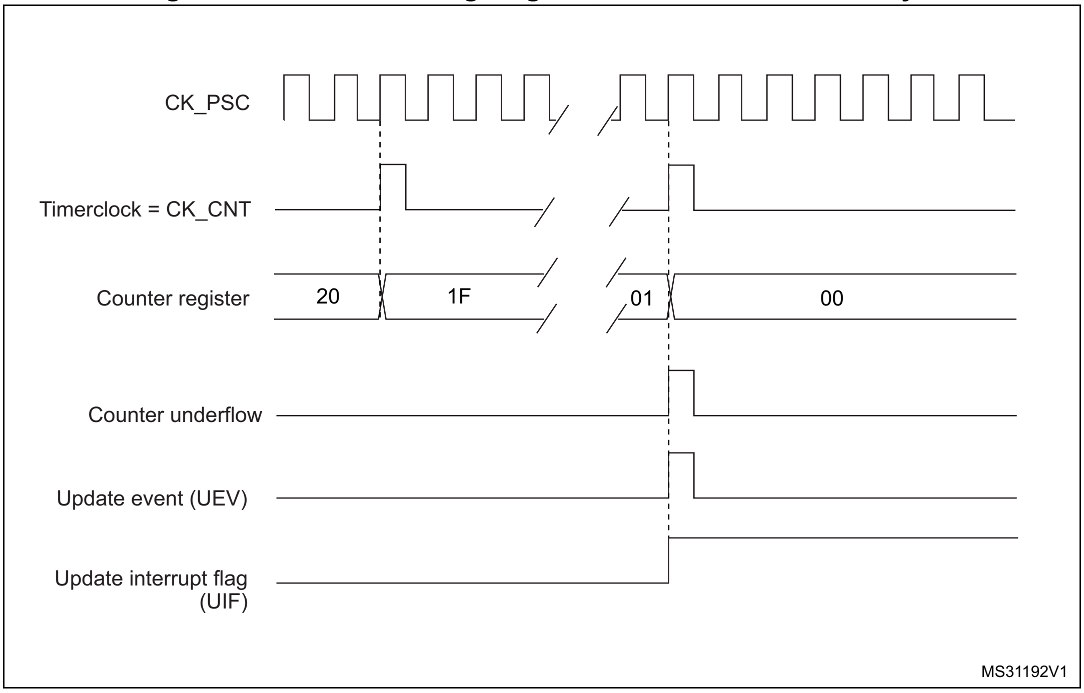


Figure 330. Counter timing diagram, update event with ARPE=1 (counter underflow)

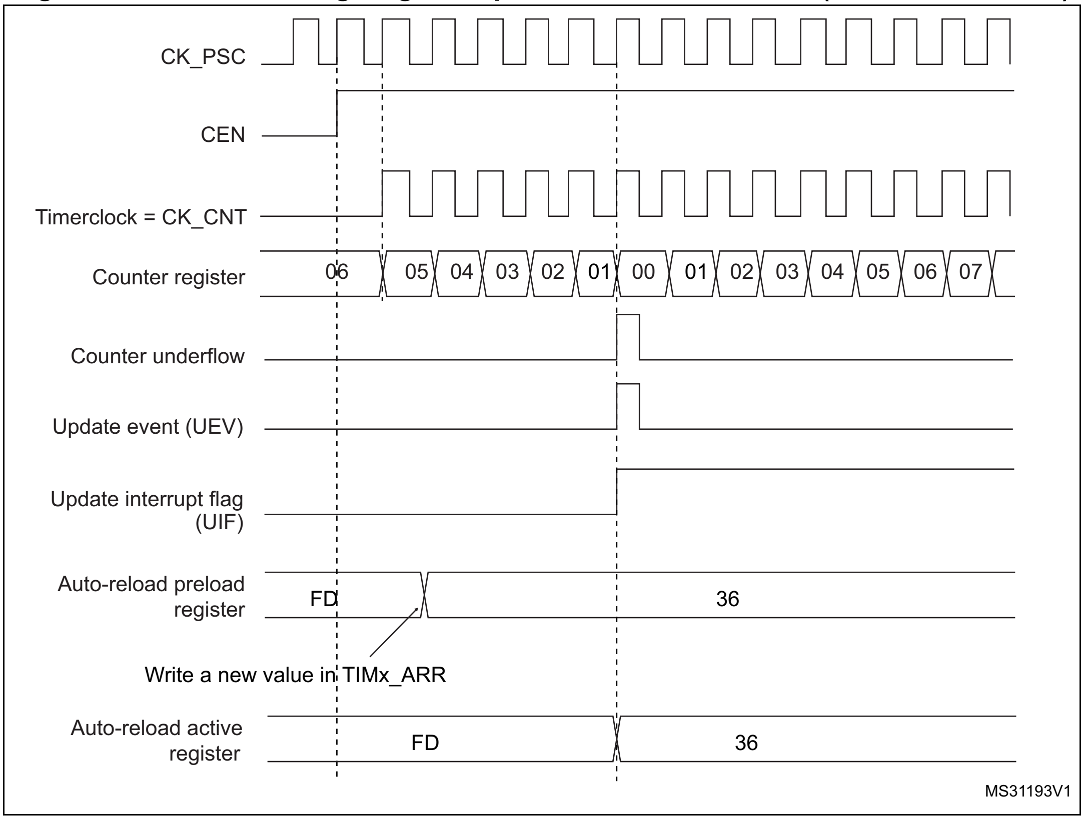
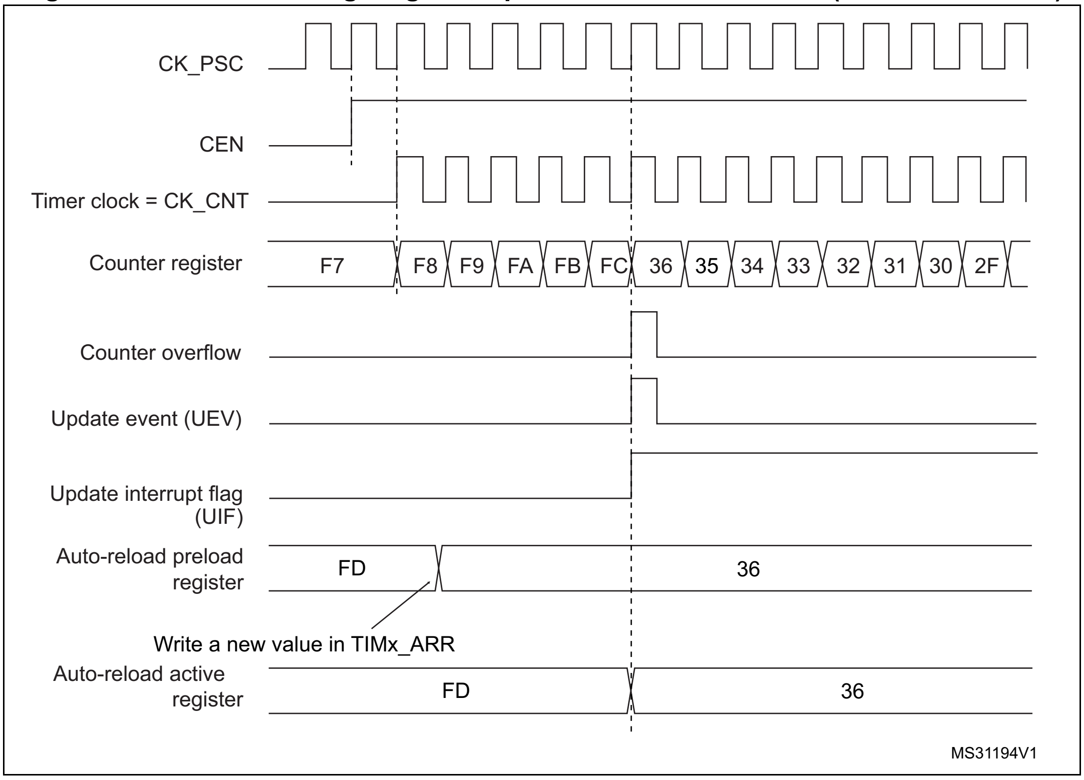


Figure 331. Counter timing diagram, Update event with ARPE=1 (counter overflow)



38.3.3 Repetition counter

Section 38.3.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

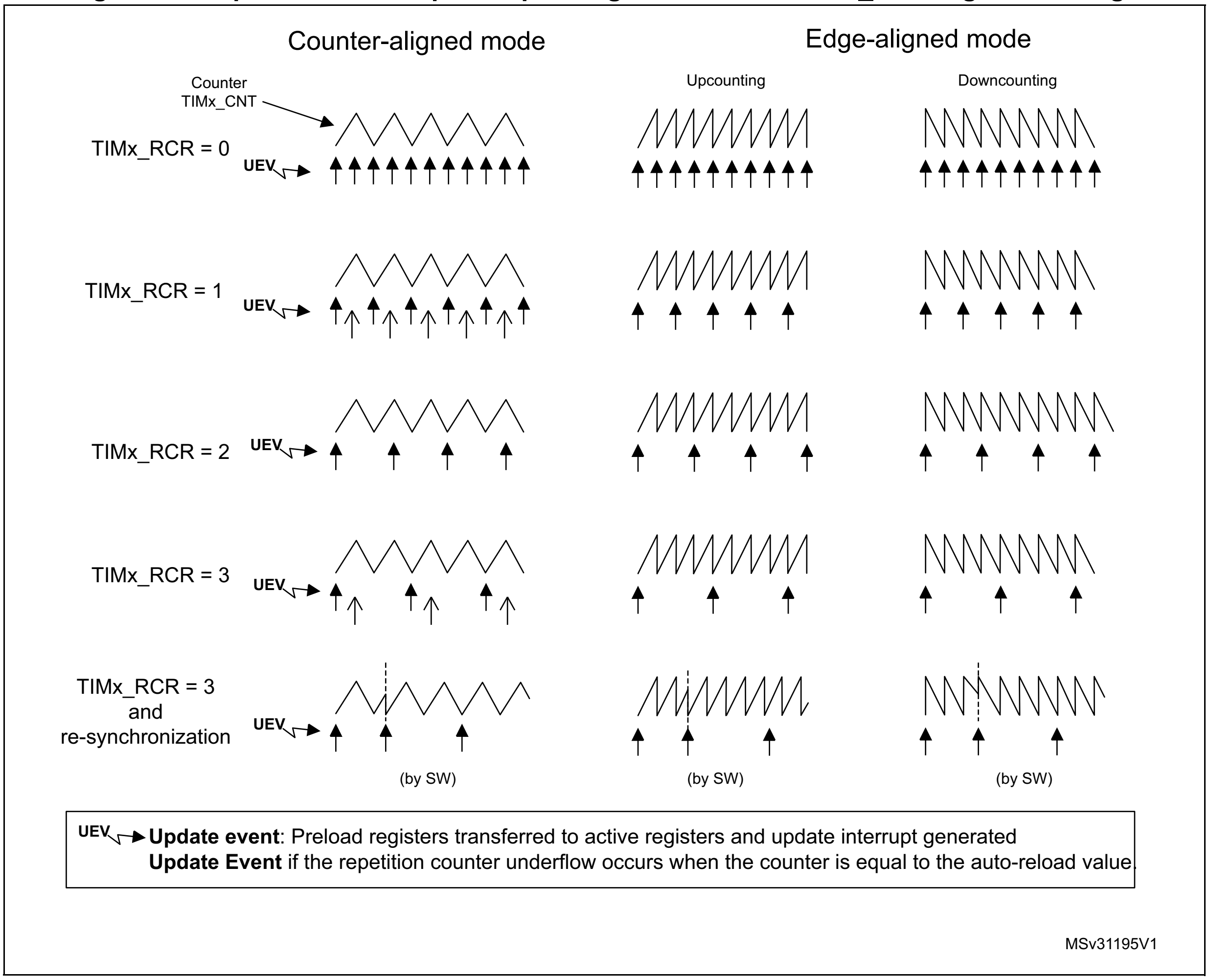
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 332*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 332. Update rate examples depending on mode and TIMx_RCR register settings



38.3.4 External trigger input

The timer features an external trigger input ETR. It can be used as:

- external clock (external clock mode 2, see *Section 38.3.5*)
- trigger for the slave mode (see *Section 38.3.26*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 38.3.7*)

Figure 333 below describes the ETR input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield.

Figure 333. External trigger input block

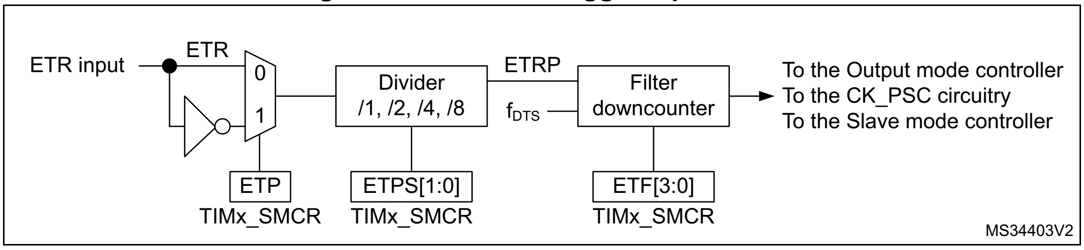
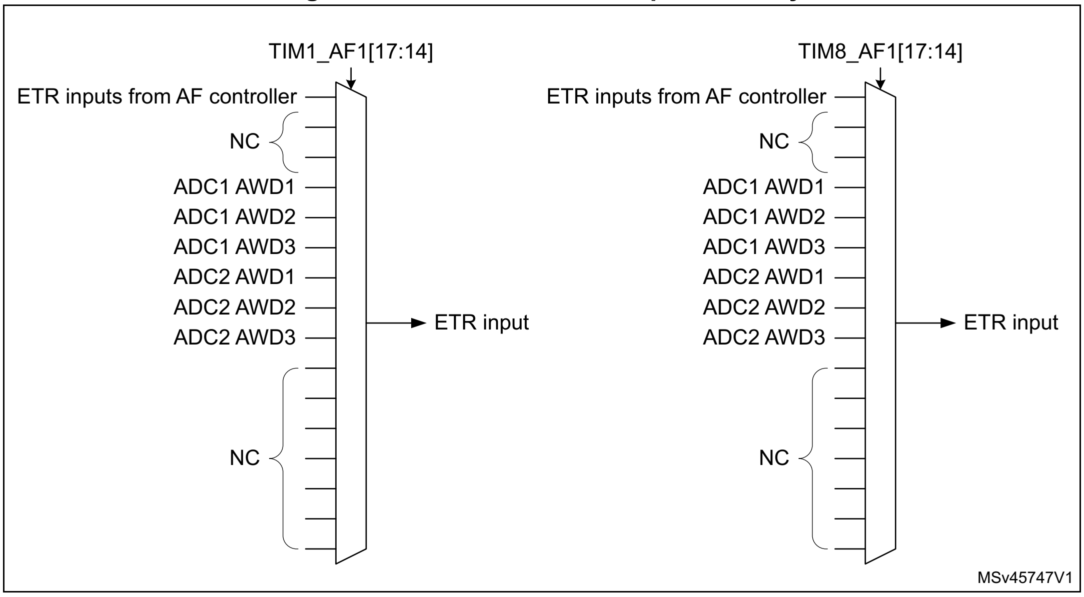


Figure 334. TIM1/TIM8 ETR input circuitry



38.3.5 Clock selection

The counter clock can be provided by the following clock sources:

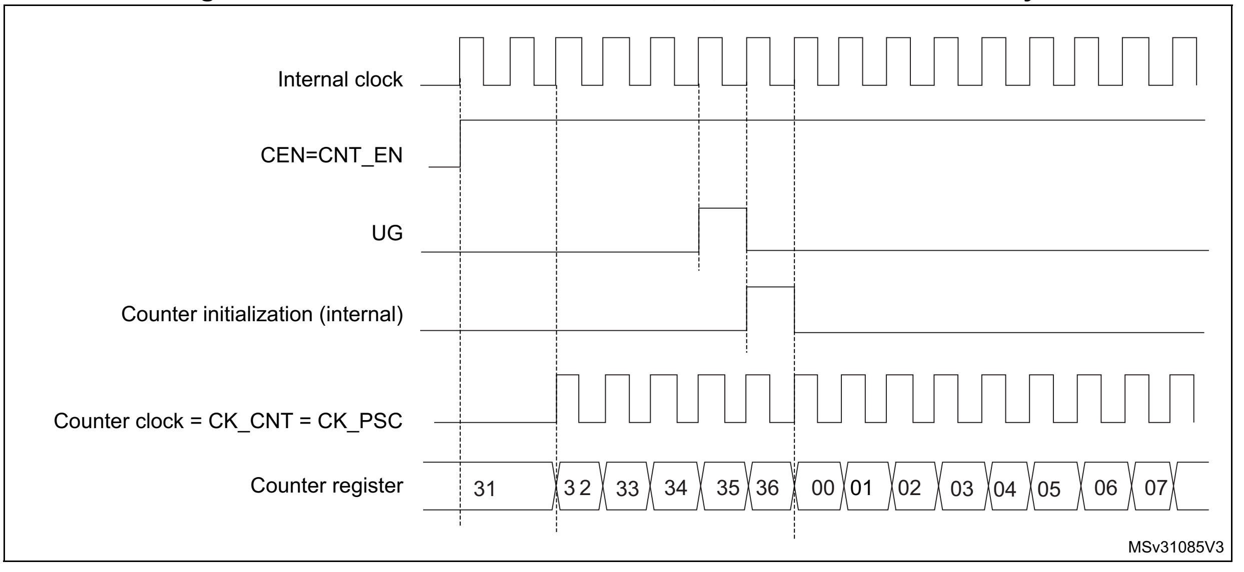
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Encoder mode

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 335 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

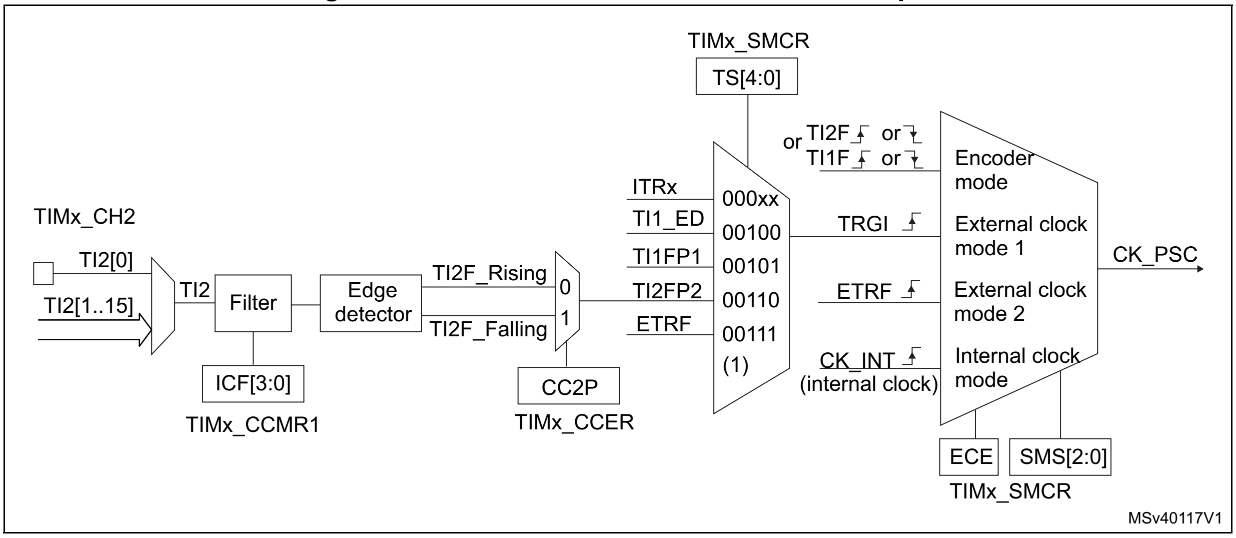
Figure 335. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 336. TI2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

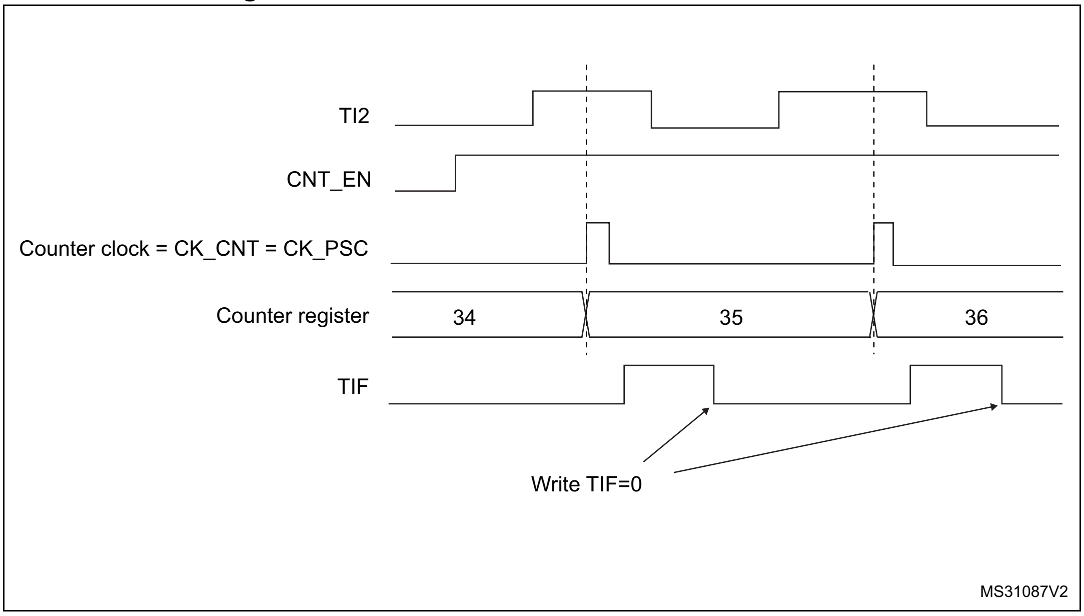
1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so the user does not need to configure it.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 337. Control circuit in external clock mode 1



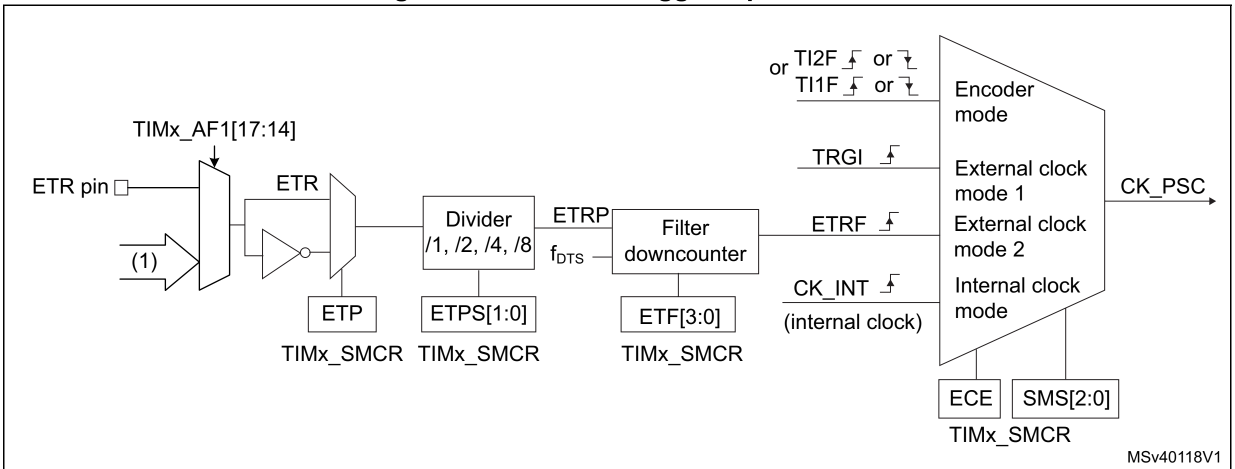
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

The *Figure 338* gives an overview of the external trigger input block.

Figure 338. External trigger input block



1. Refer to *Figure 334: TIM1/TIM8 ETR input circuitry*.

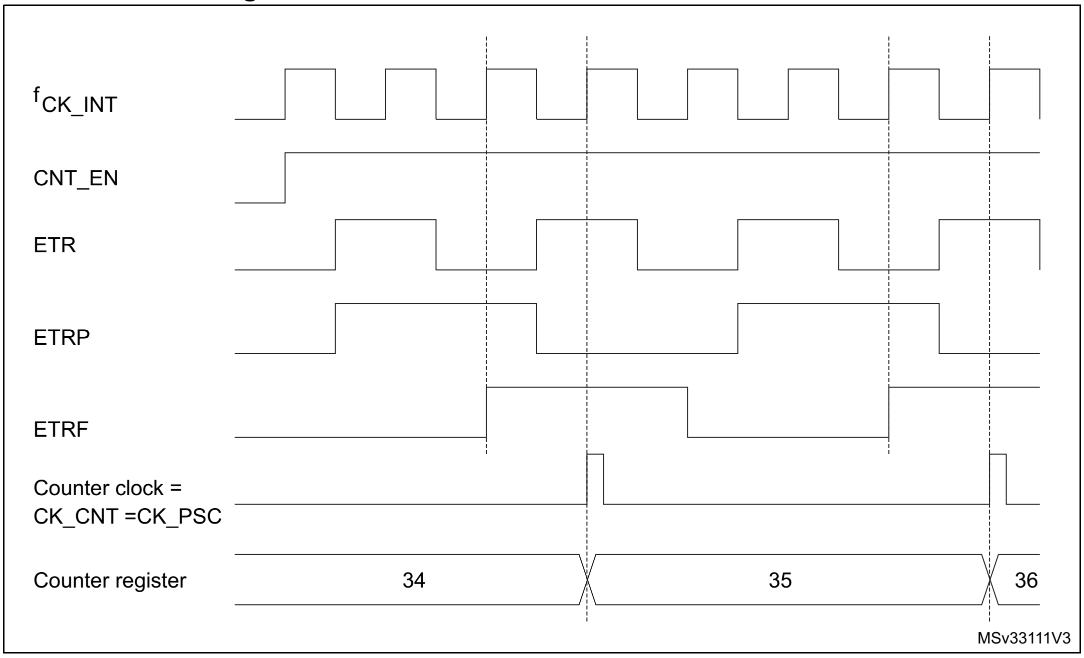
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by proper ETPS prescaler setting.

Figure 339. Control circuit in external clock mode 2



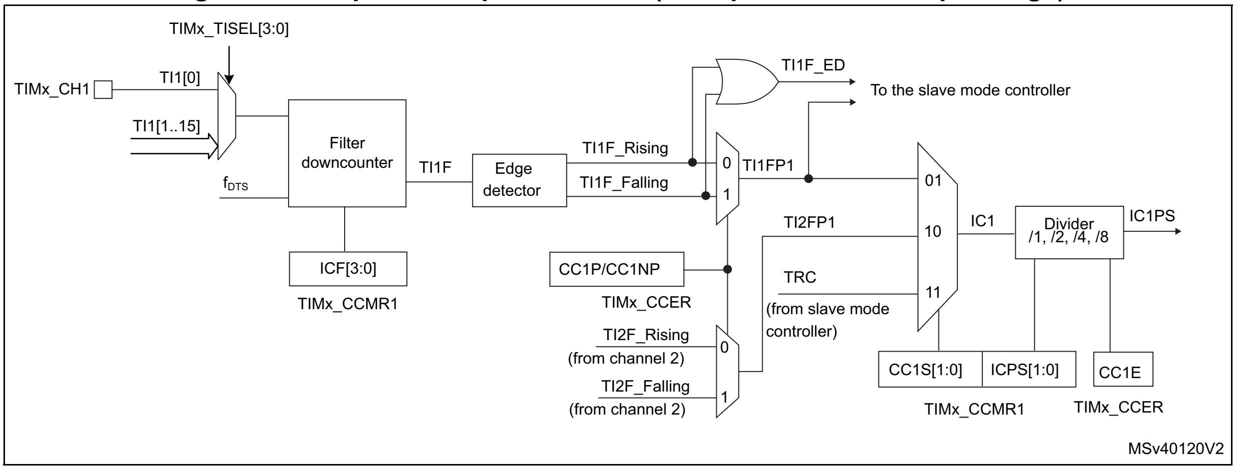
38.3.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 340 to *Figure 343* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 340. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 341. Capture/compare channel 1 main circuit

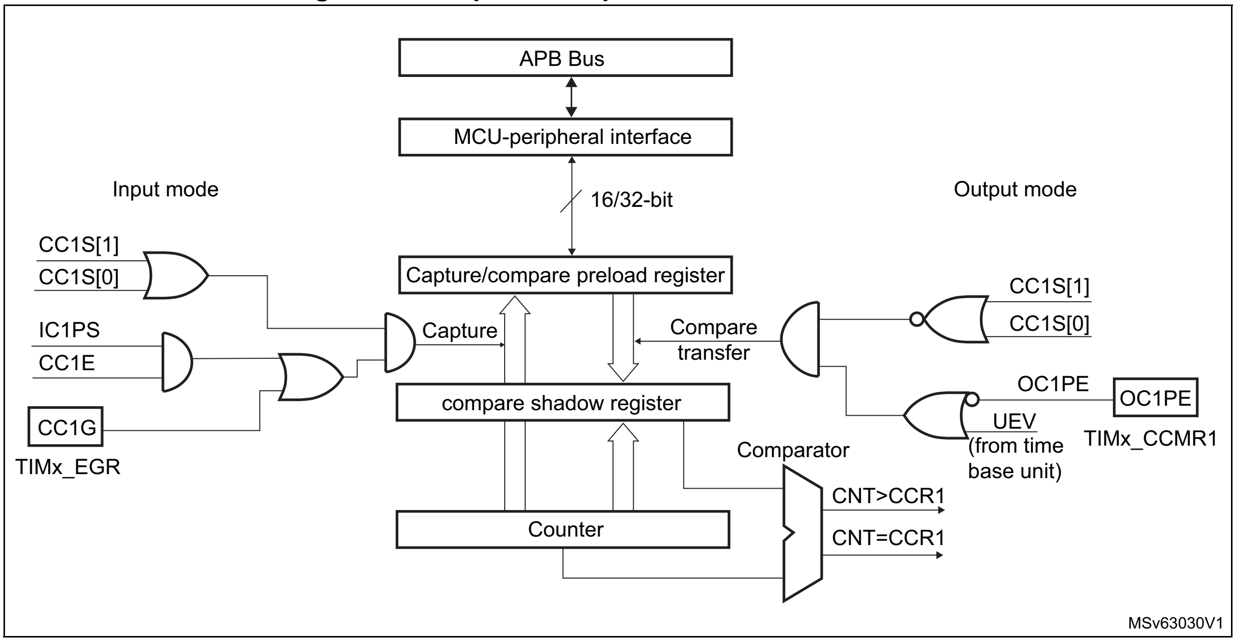
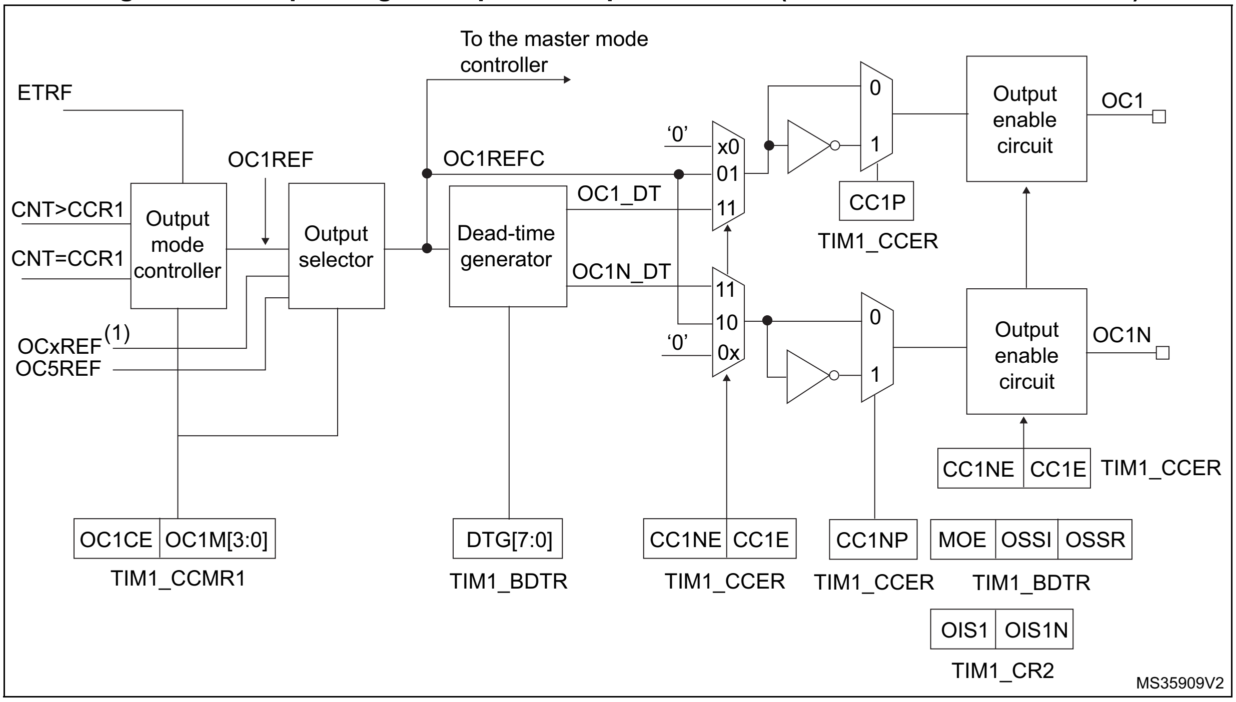


Figure 342. Output stage of capture/compare channel (channel 1, idem ch. 2 and 3)



1. OCxREF, where x is the rank of the complementary channel

Figure 343. Output stage of capture/compare channel (channel 4)

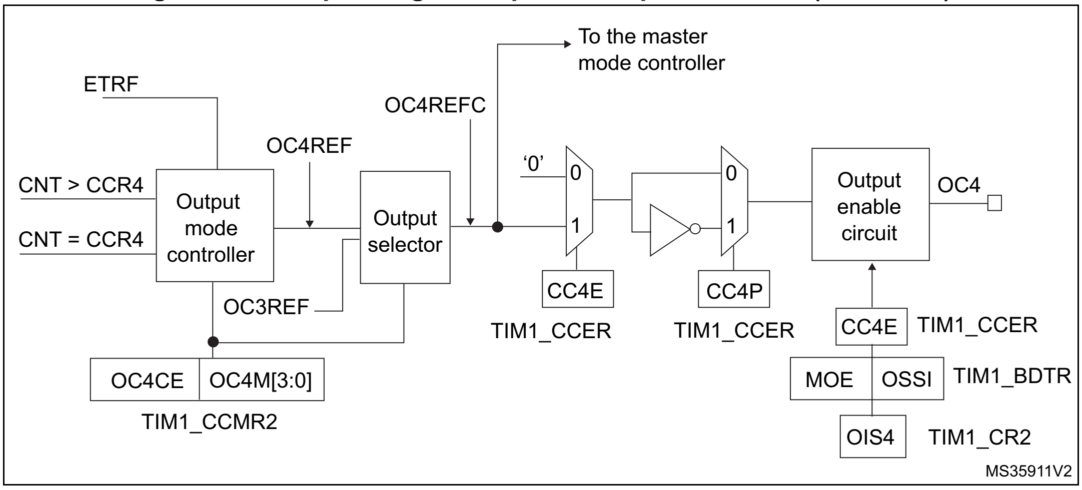
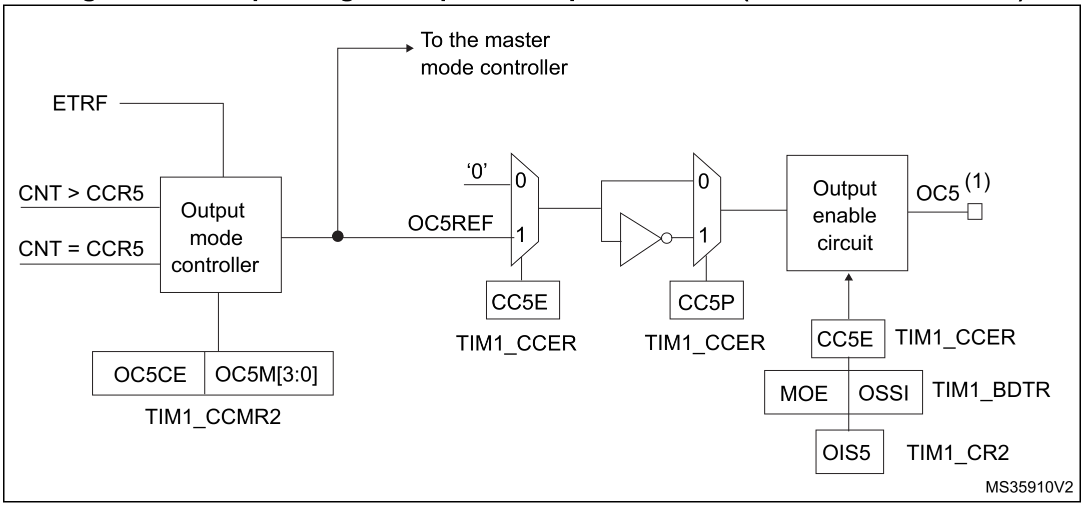


Figure 344. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

38.3.7 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when written with '0'.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been

detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

38.3.8 PWM input mode

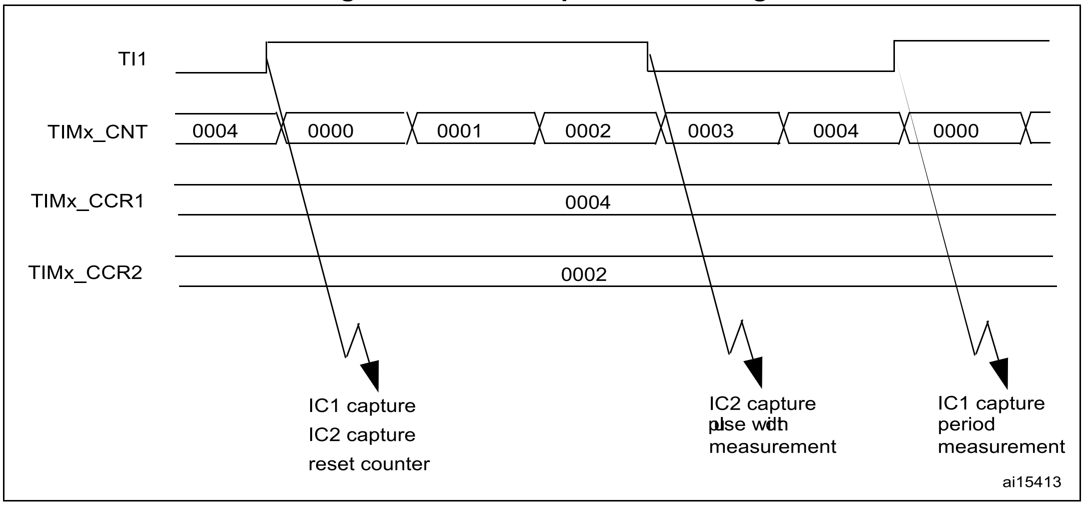
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, the user can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP=’10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 345. PWM input mode timing



38.3.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

38.3.10 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while Channel 5 and 6 are only available inside the device (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=0000), be set active (OCxM=0001), be set inactive (OCxM=0010) or can toggle (OCxM=0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

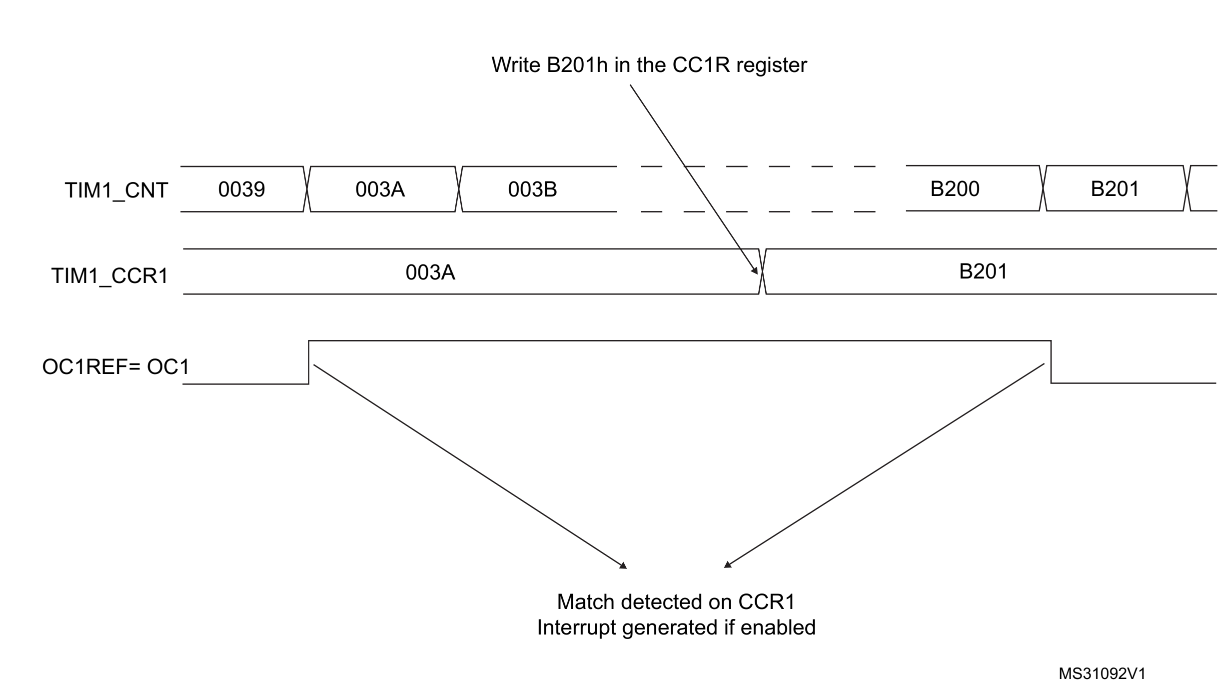
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   – Write OCxM = 0011 to toggle OCx output pin when CNT matches CCRx
   – Write OCxPE = 0 to disable preload register
   – Write CCxP = 0 to select active high polarity
   – Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 346*.

Figure 346. Output compare mode, toggle on OC1



38.3.11 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘0110’ (PWM mode 1) or ‘0111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

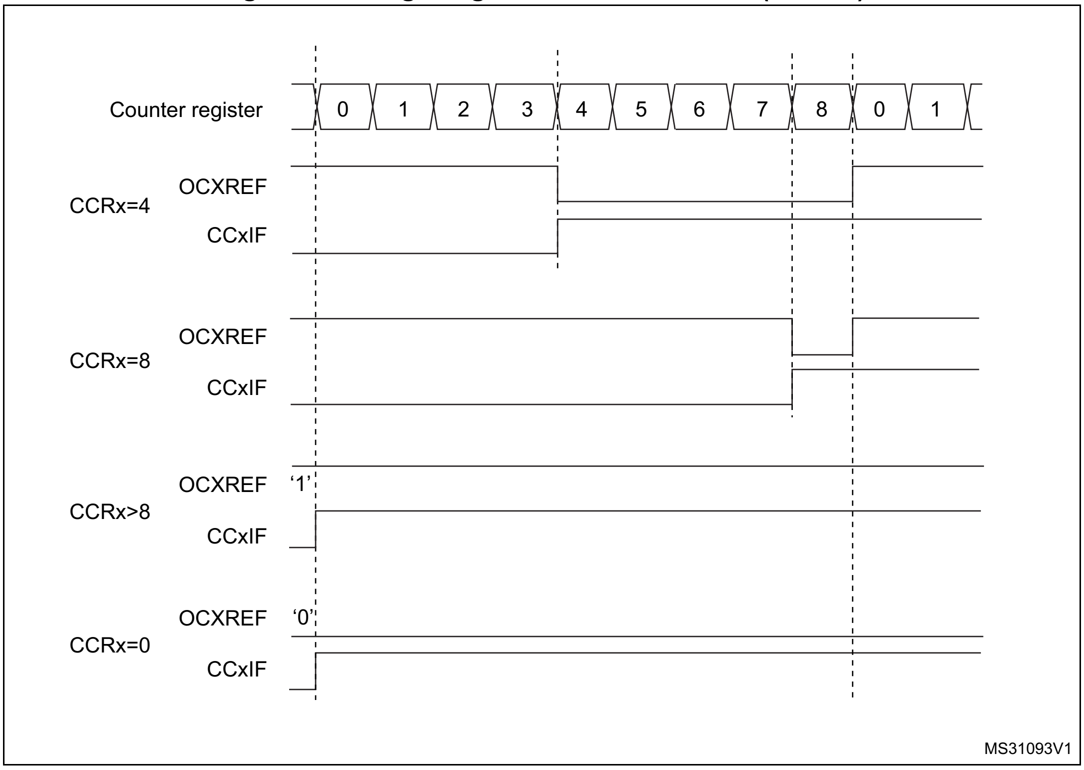
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to the *Upcounting mode on page 1905*.

  In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1’. If the compare value is 0 then OCxRef is held at ‘0’. *Figure 347* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 347. Edge-aligned PWM waveforms (ARR=8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode on page 1909*

  In PWM mode 1, the reference signal OCxRef is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then OCxREF is held at ‘1’. 0% PWM is not possible in this mode.

PWM center-aligned mode

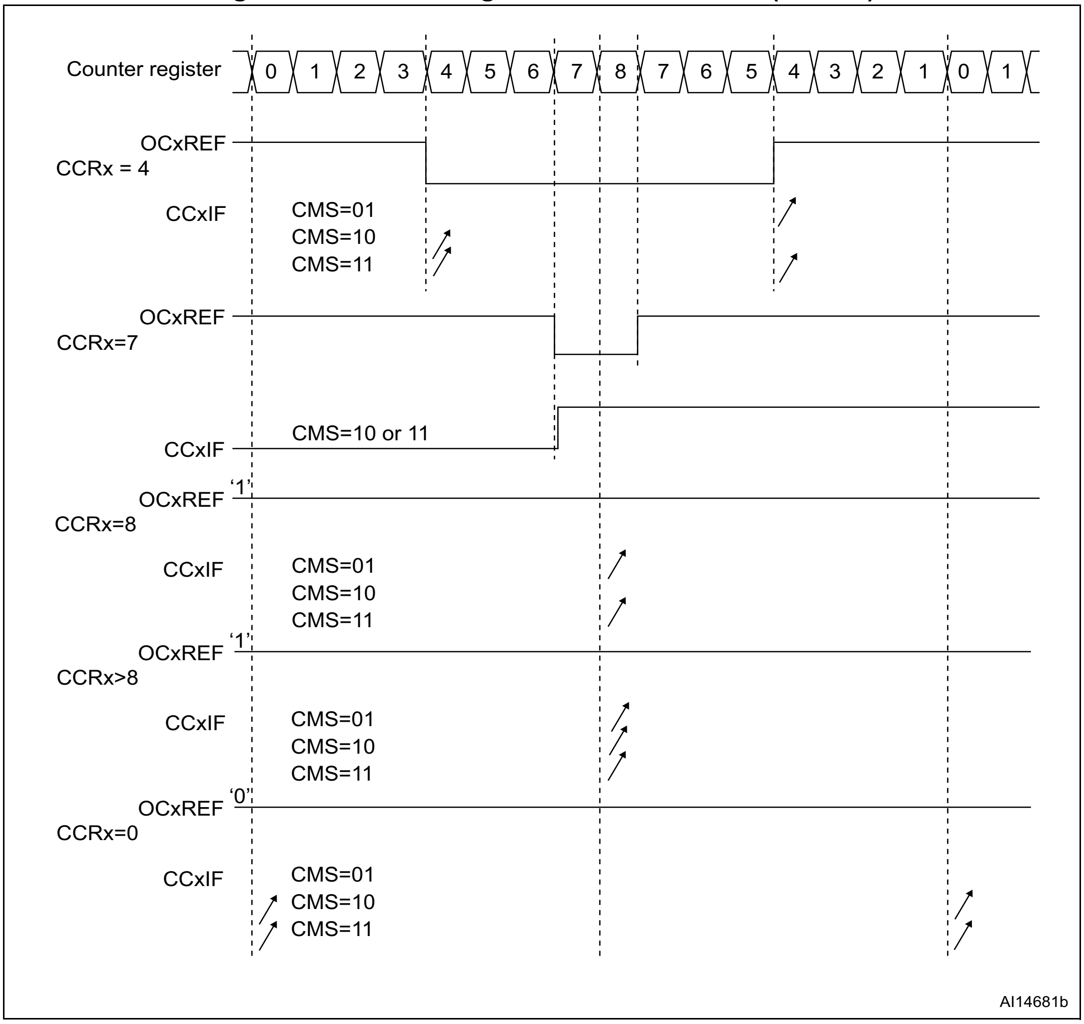
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the

TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to the *Center-aligned mode (up/down counting) on page 1912*.

Figure 348 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 348. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

38.3.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

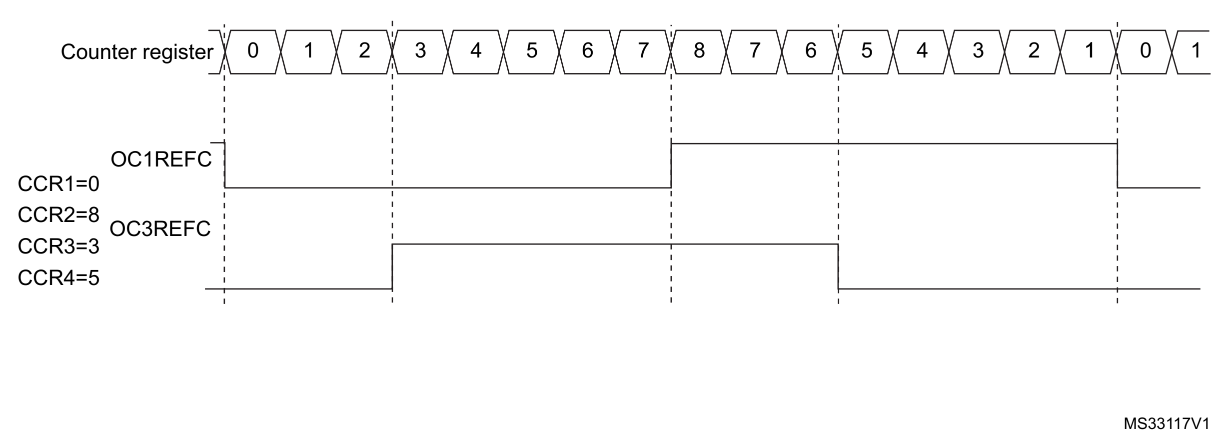
Asymmetric PWM mode can be selected independently on two channel (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 1.

Figure 349 represents an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 349. Generation of 2 phase-shifted PWM signals with 50% duty cycle



38.3.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

– OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
– OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

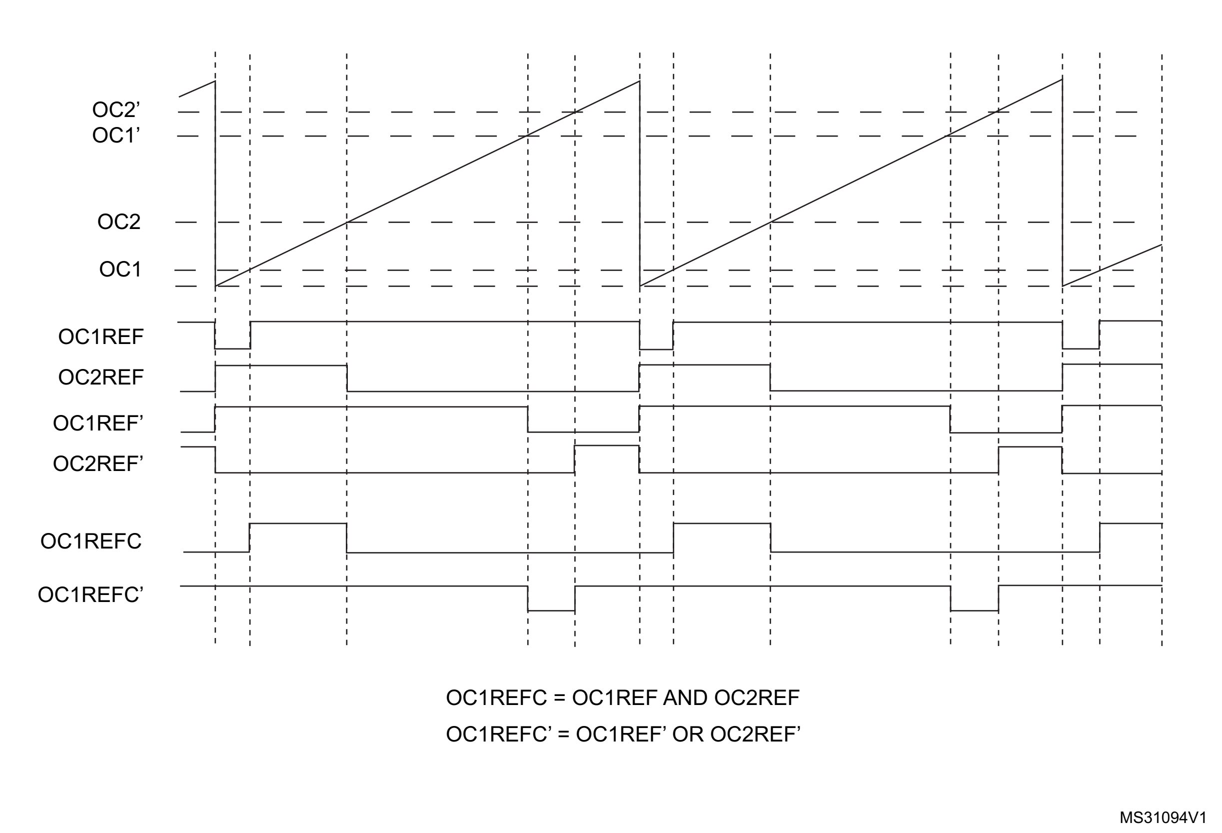
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 350 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

– Channel 1 is configured in Combined PWM mode 2,
– Channel 2 is configured in PWM mode 1,
– Channel 3 is configured in Combined PWM mode 2,
– Channel 4 is configured in PWM mode 1.

Figure 350. Combined PWM mode on channel 1 and 3



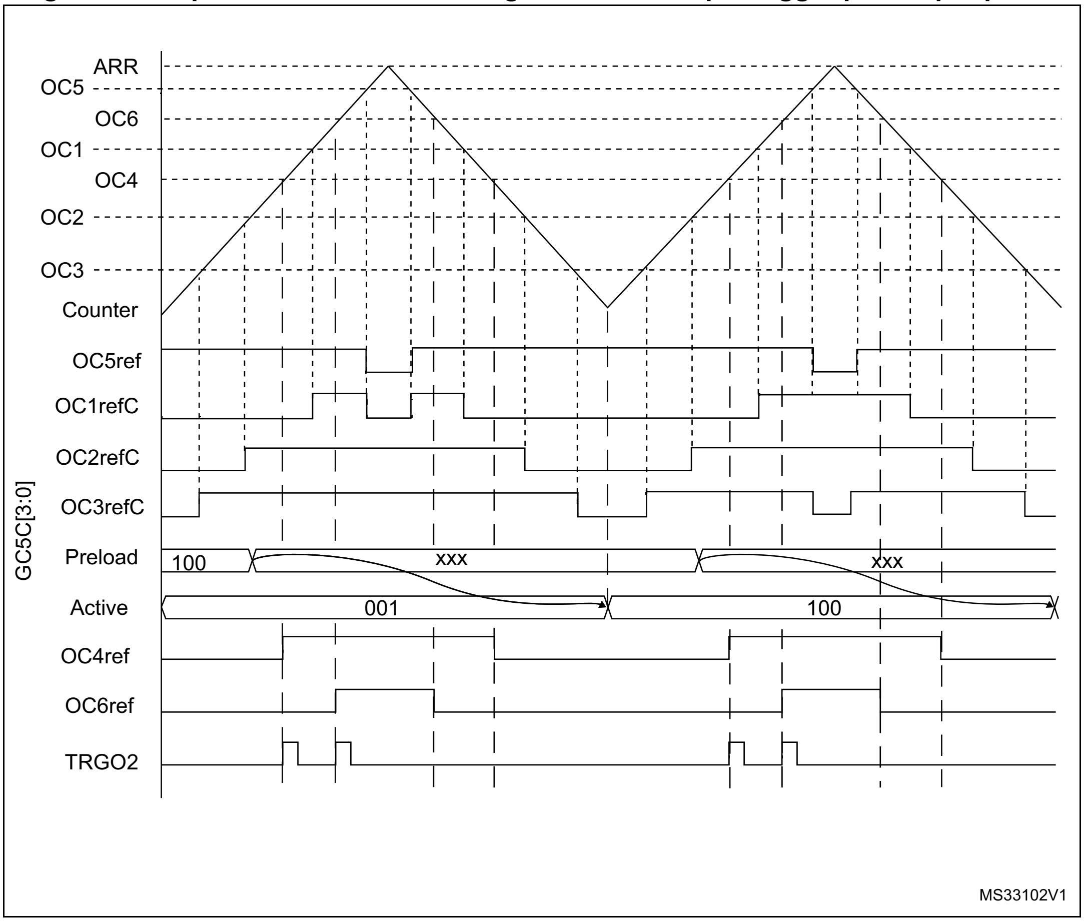
38.3.14 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The OC5REF signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the OC5REF is combined. The resulting signals, OCxREFC, are made of an AND logical combination of two reference PWMs:

– If GC5C1 is set, OC1REFC is controlled by TIMx_CCR1 and TIMx_CCR5
– If GC5C2 is set, OC2REFC is controlled by TIMx_CCR2 and TIMx_CCR5
– If GC5C3 is set, OC3REFC is controlled by TIMx_CCR3 and TIMx_CCR5

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 351. 3-phase combined PWM signals with multiple trigger pulses per period



The TRGO2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 38.3.27: ADC synchronization* for more details.

38.3.15 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 279: Output control bits for complementary OCx and OCxN channels with break feature on page 1981* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 352. Complementary output with dead-time insertion

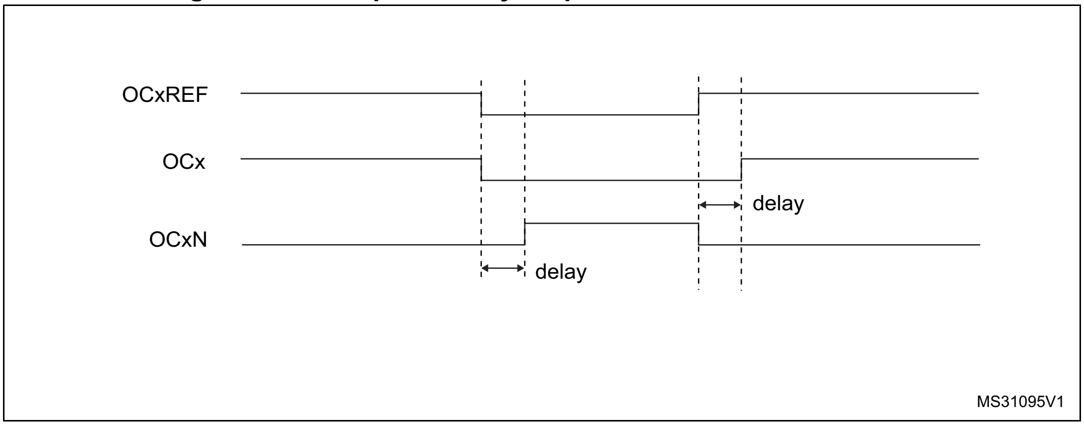


Figure 353. Dead-time waveforms with delay greater than the negative pulse

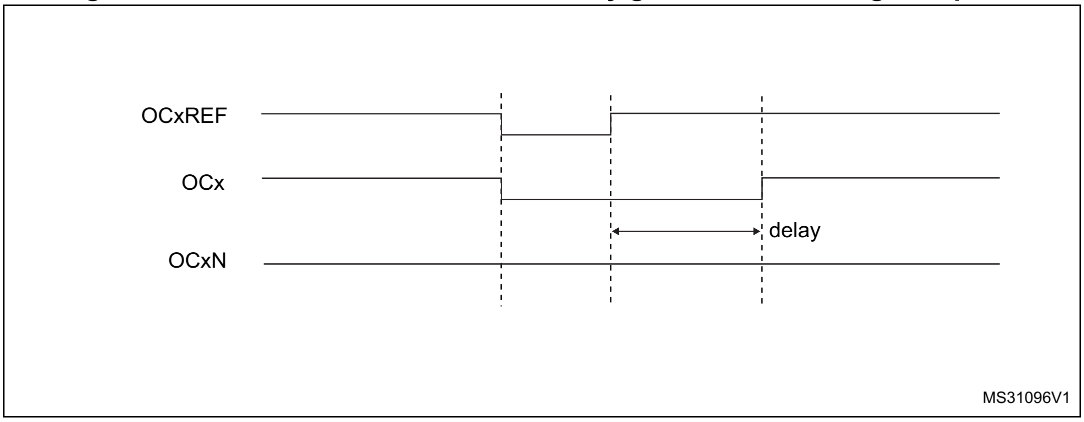
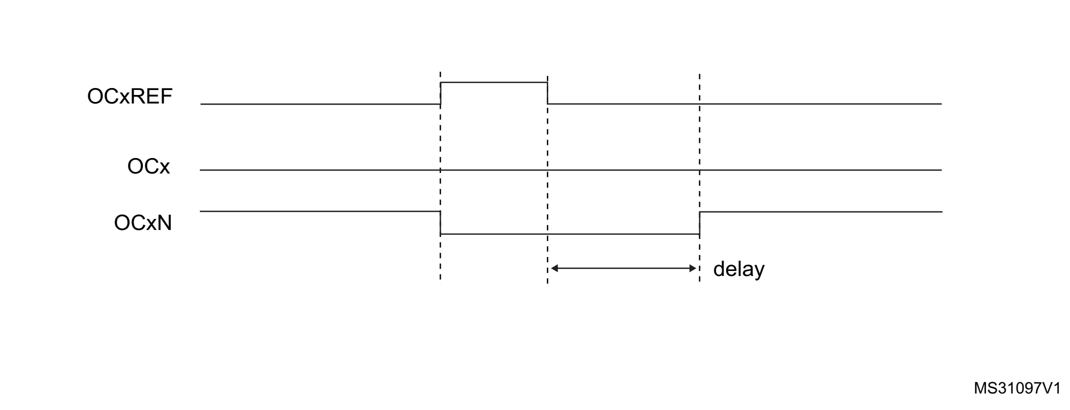


Figure 354. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 38.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

38.3.16 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM1 and TIM8 timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows the outputs to be enabled/disabled by software and is reset in case of break or break2 event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 279: Output control bits for complementary OCx and OCxN channels with break feature on page 1981* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKE/BK2E and BKP/BK2P can be modified at the same time. When the BKE/BK2E and BKP/BK2P bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source:
  - the analog watchdog output of the DFSDM1 peripheral
  - A system break:

    - the Cortex®-M4 LOCKUP output

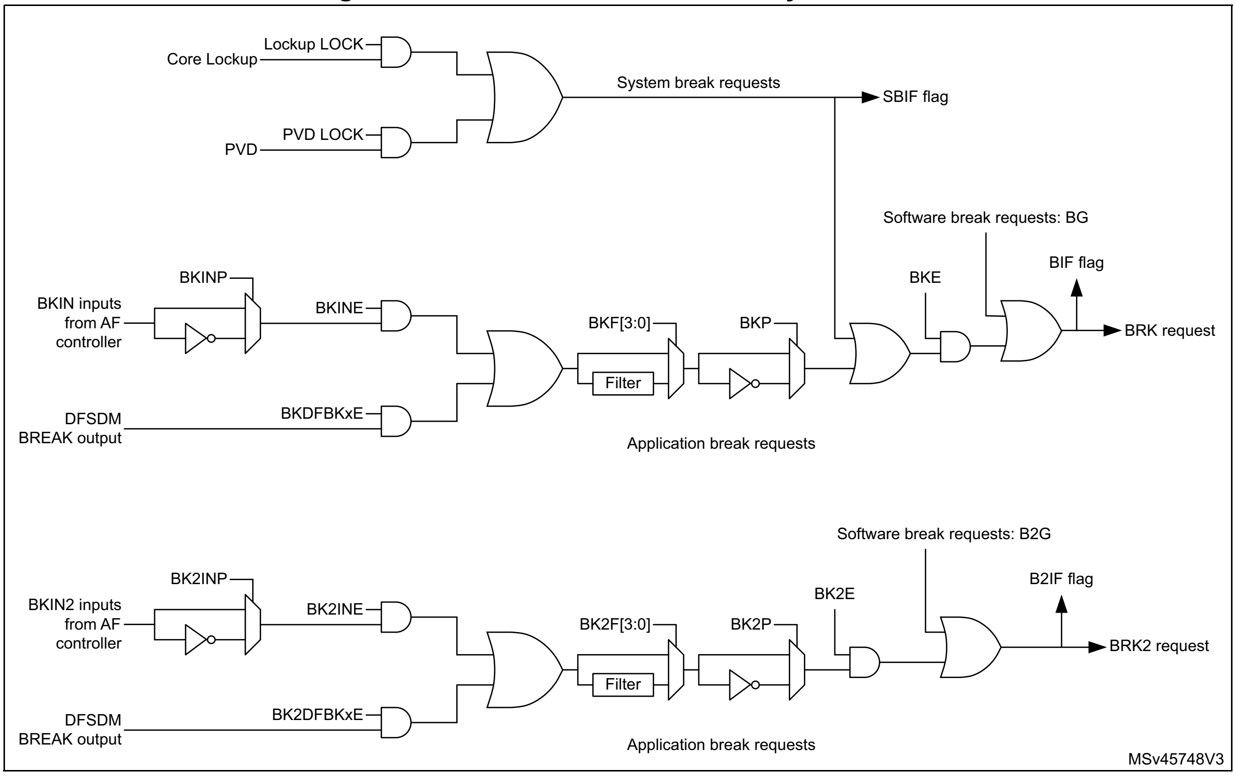
    - the PVD output

The source for break2 (BRK2) is an external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering.

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register. The software break generation using BG and B2G is active whatever the BKE and BK2E enable bits values.

All sources are ORed before entering the timer BRK or BRK2 inputs, as per *Figure 355* below.

Figure 355. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their

active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 ck_tim clock cycles).

– If OSSI=0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

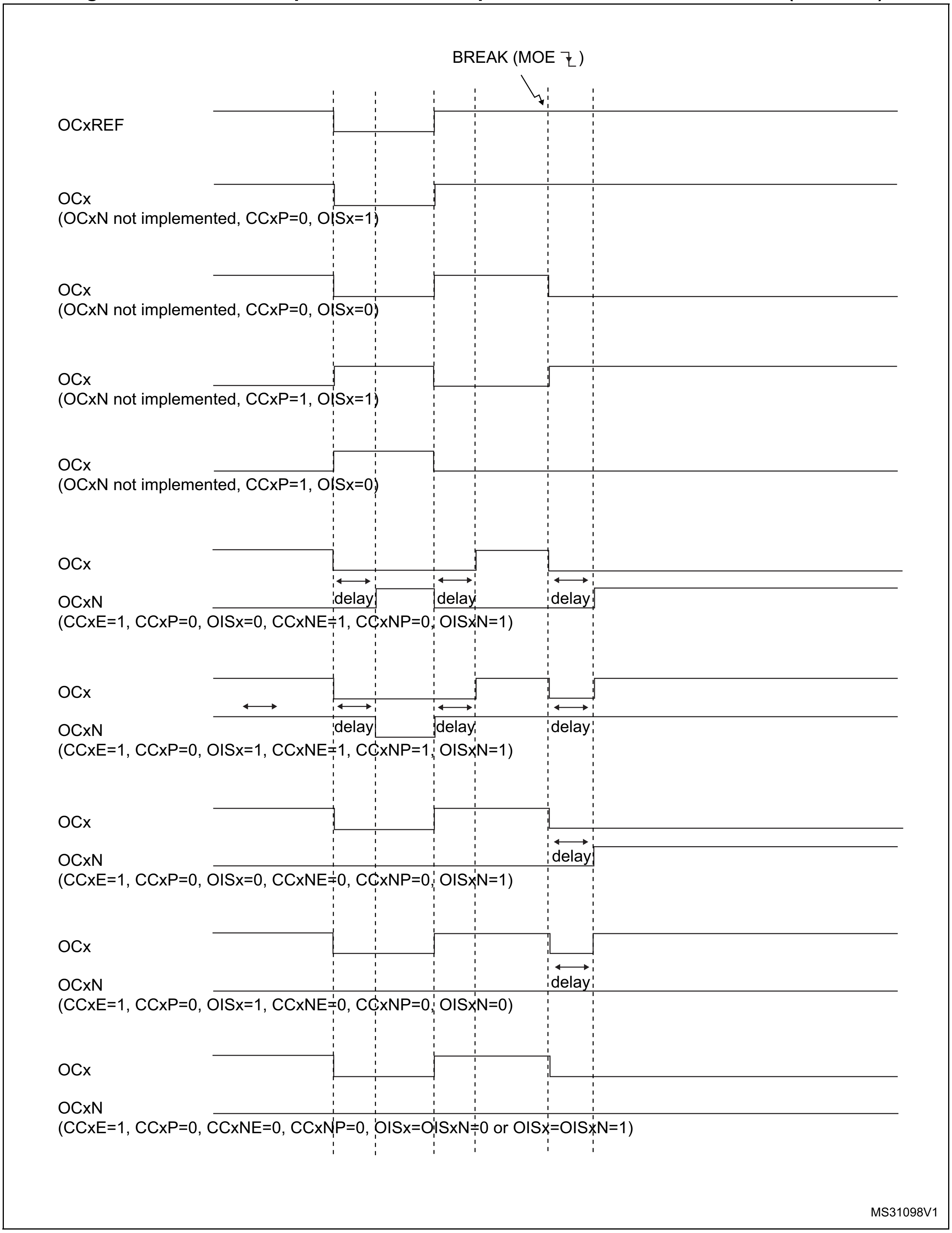
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.
If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

Note: The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The application can choose from 3 levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 38.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*. The LOCK bits can be written only once after an MCU reset.

Figure 356 shows an example of behavior of the outputs in response to a break.

Figure 356. Various output behavior in response to a break event on BRK (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The BRK input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- BRK2 can only disable (inactive state) the PWM outputs.

The BRK has a higher priority than BRK2 input, as described in Table 275.

Note: BRK2 must only be used with OSSR = OSSI = 1.

Table 275. Behavior of timer outputs versus BRK/BRK2 inputs

BRK	BRK2	Timer outputs state	Typical use case	
			OCxN output (low side switches)	OCx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 357 gives an example of OCx and OCxN output behavior in case of active signals on BRK and BRK2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 357. PWM output state following BRK and BRK2 pins assertion (OSSI=1)

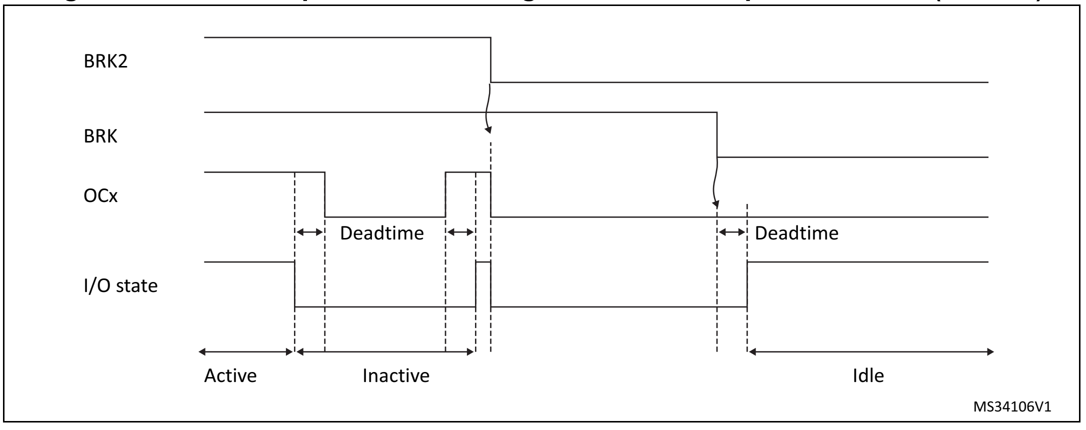
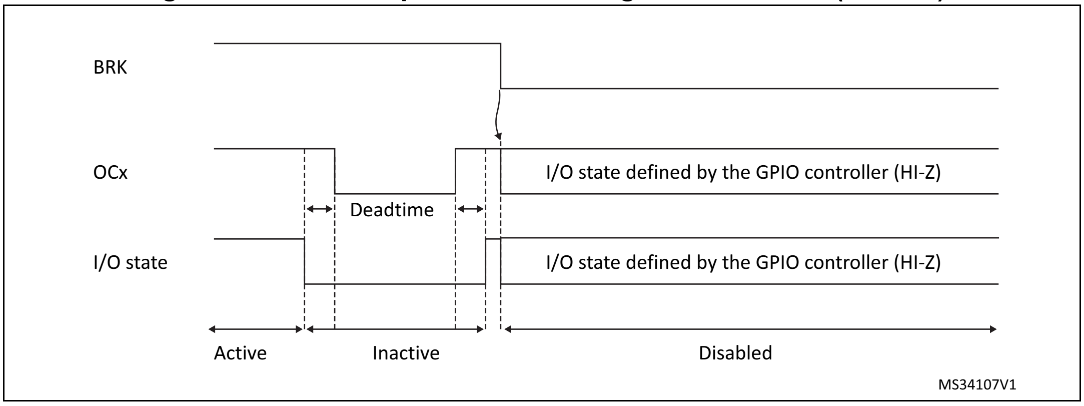


Figure 358. PWM output state following BRK assertion (OSSI=0)



38.3.17 Bidirectional break inputs

The TIM1/TIM8 are featuring bidirectional break I/Os, as represented on *Figure 359*.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break and break2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the break and break2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BK(2)E = 1). When a software break event is generated with BK(2)E = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break(2) I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BK(2)DSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 276*)

Table 276. Break protection disarming conditions

MOE	BKDIR (BK2DIR)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

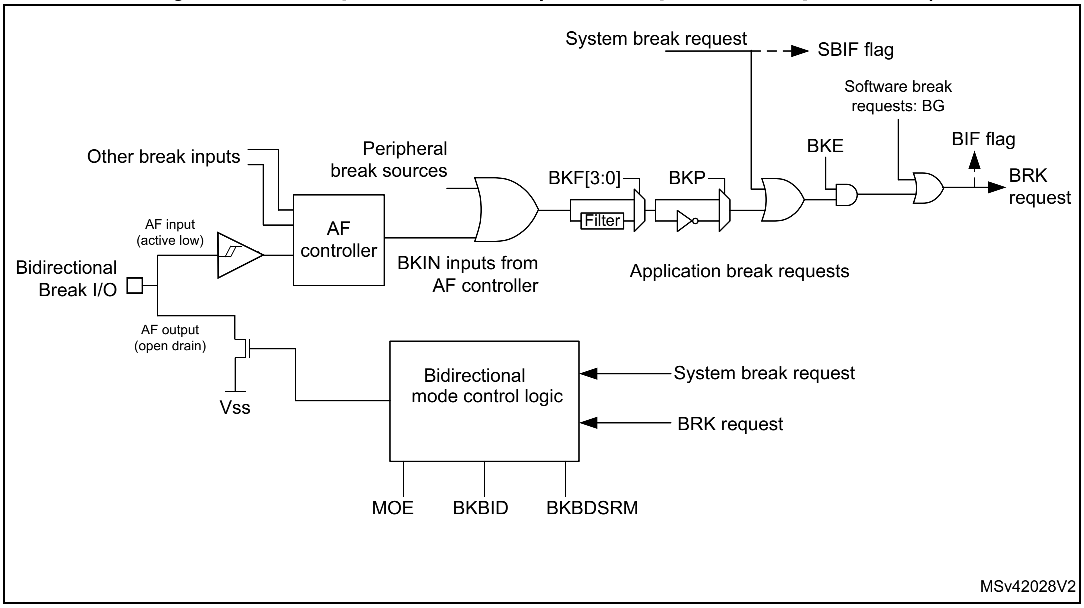
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 359. Output redirection (BRK2 request not represented)



38.3.18 Clearing the OCxREF signal on an external event

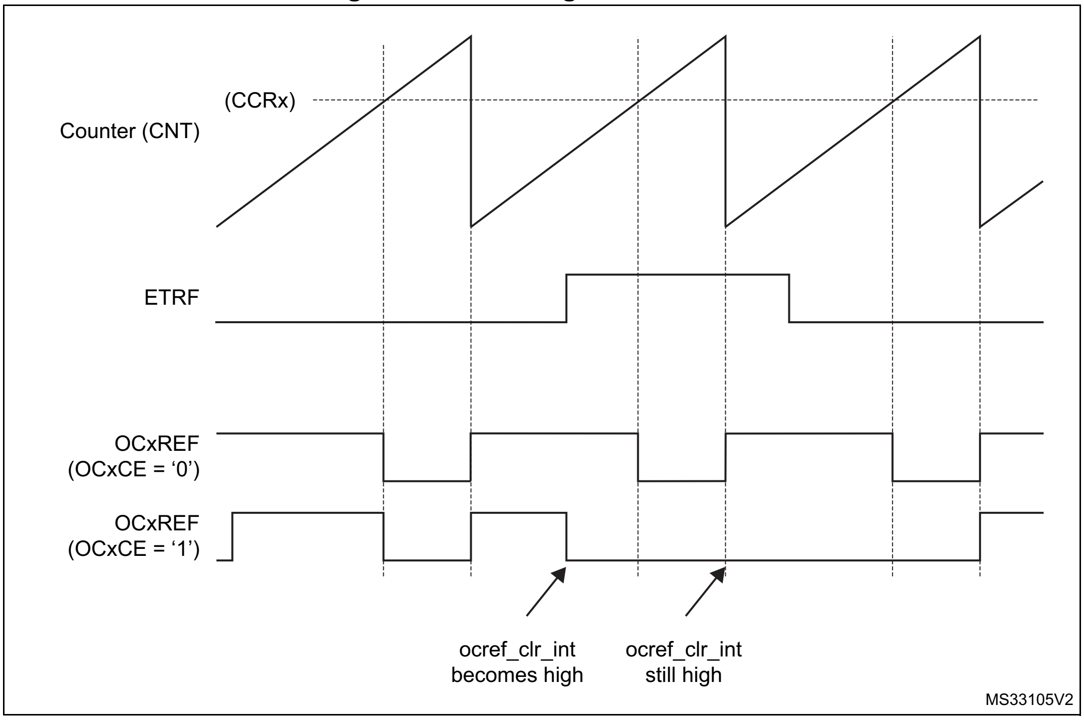
The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. The ocref_clr_int is connected to the ETRF signal (ETR after filtering).

When ETRF is chosen, ETR must be configured as follows:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

Figure 360 shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 360. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then OCxREF is enabled again at the next counter overflow.

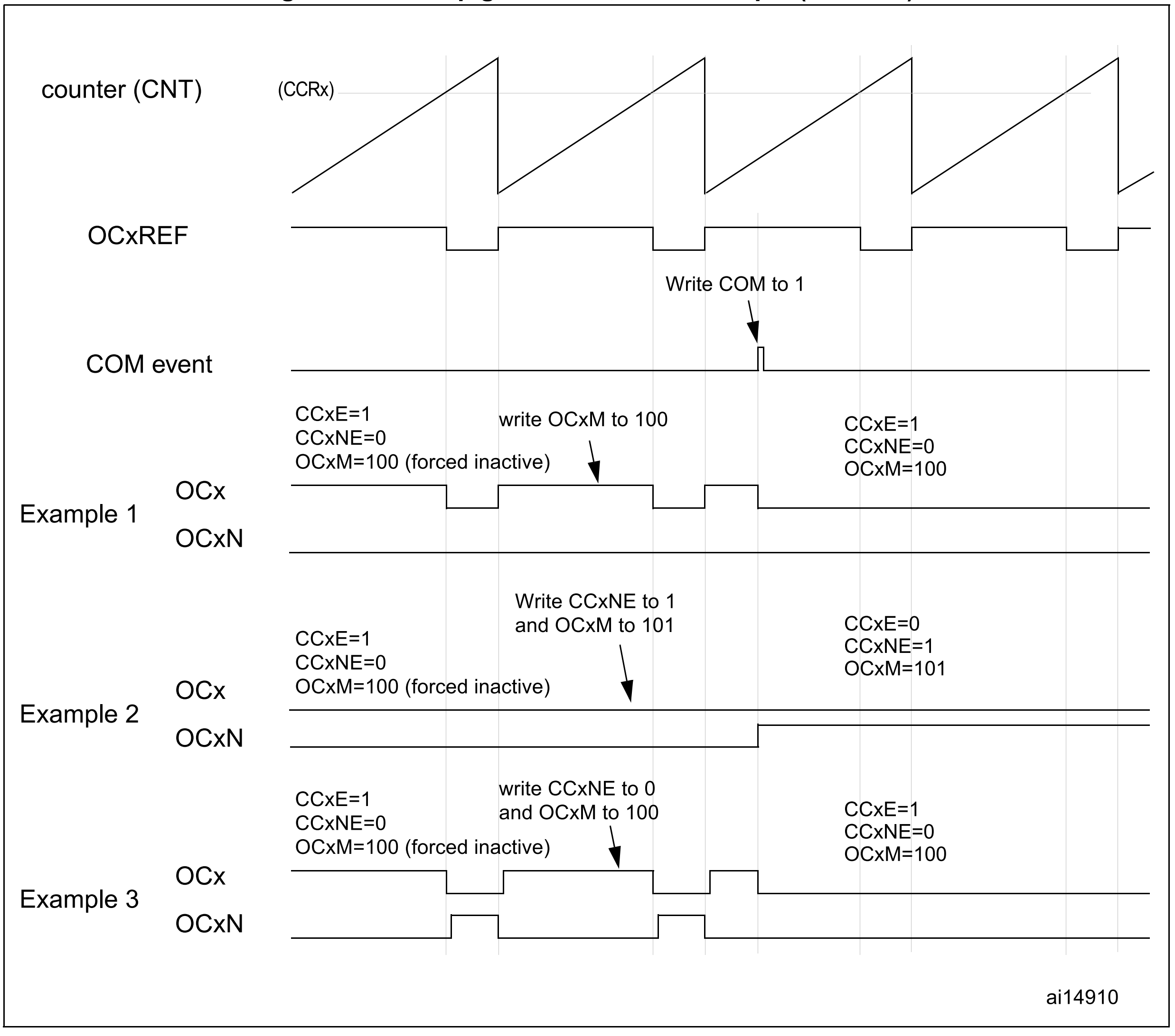
38.3.19 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 361* describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 361. 6-step generation, COM example (OSSR=1)



38.3.20 One-pulse mode

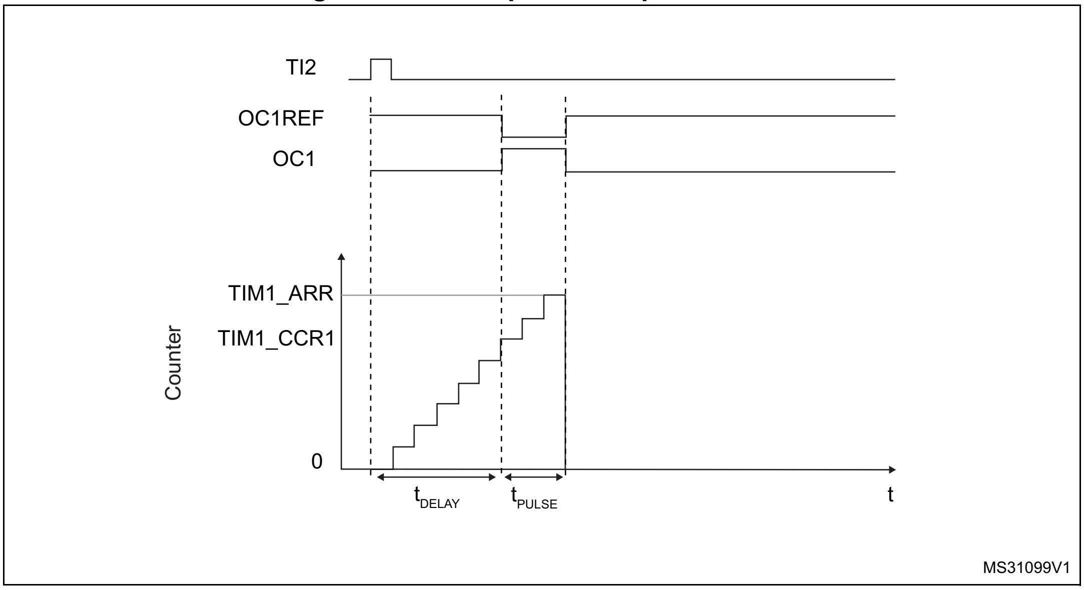
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 362. Example of one pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

38.3.21 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 38.3.20*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

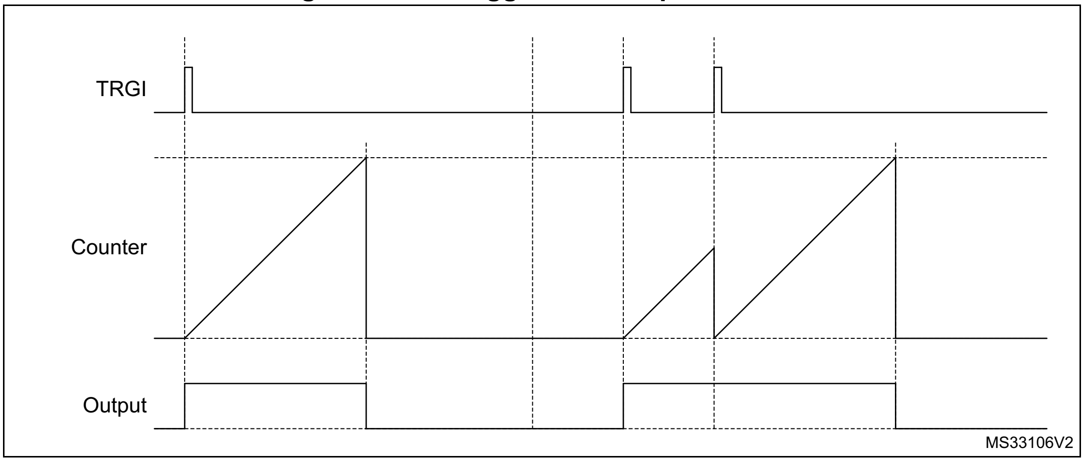
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 363. Retriggerable one pulse mode



38.3.22 Encoder interface mode

To select Encoder Interface mode write SMS='001' in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS='010' if it is counting on TI1 edges only and SMS='011' if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to a quadrature encoder. Refer to *Table 277*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1'). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

Note: The prescaler must be set to zero when encoder mode is enabled

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 277. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

The *Figure 364* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (TIMx_CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (TIMx_CCMR1 register, TI2FP2 mapped on TI2)
- CC1P='0' and CC1NP='0' (TIMx_CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0' and CC2NP='0' (TIMx_CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (TIMx_CR1 register, Counter enabled).

Figure 364. Example of counter operation in encoder interface mode.

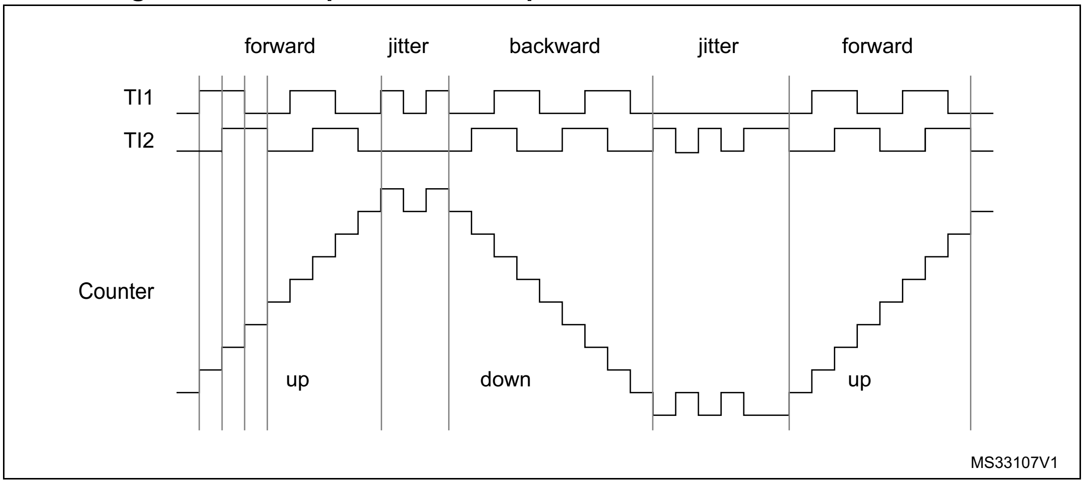
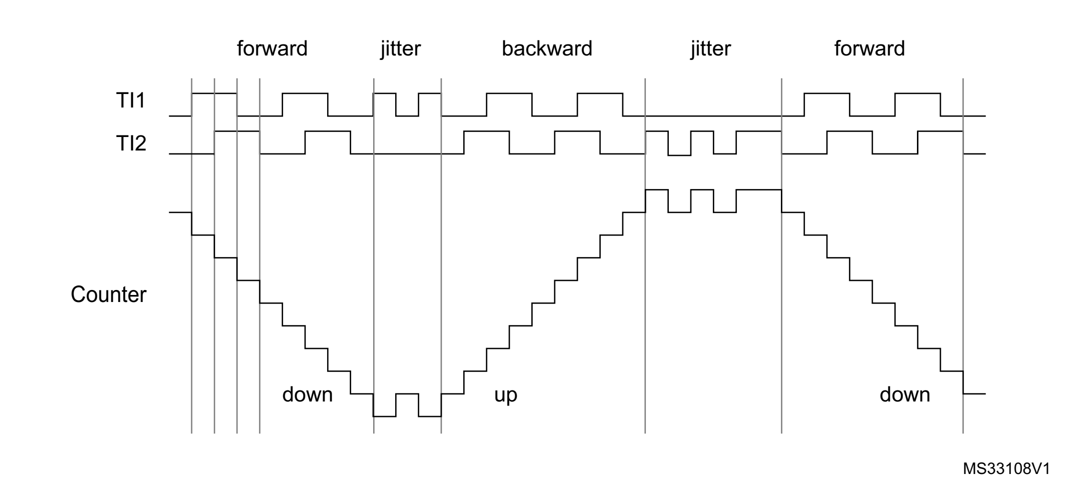


Figure 365 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 365. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

38.3.23 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

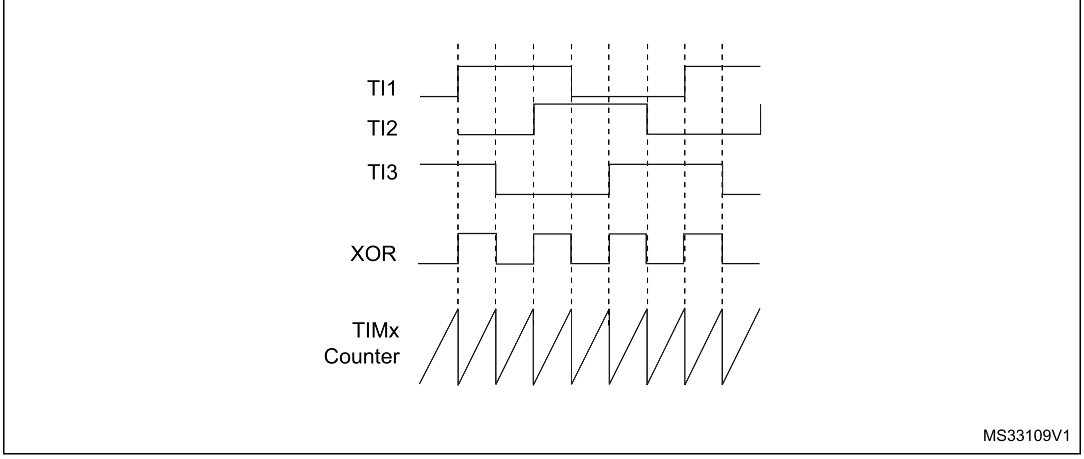
There is no latency between the UIF and UIFCPY flags assertion.

38.3.24 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMx_CH1, TIMx_CH2 and TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 366* below.

Figure 366. Measuring time interval between edges on 3 signals



38.3.25 Interfacing with Hall sensors

This is done using the advanced-control timers (TIM1 or TIM8) to generate PWM signals to drive the motor and another timer TIMx (TIM2, TIM3, TIM4) referred to as “interfacing timer” in *Figure 367*. The “interfacing timer” captures the 3 timer input pins (CC1, CC2, CC3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (See *Figure 340: Capture/compare channel (example: channel 1 input stage) on page 1923*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (TIM1 or TIM8) (by triggering a COM event). The TIM1 timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1 or TIM8) through the TRGO output.

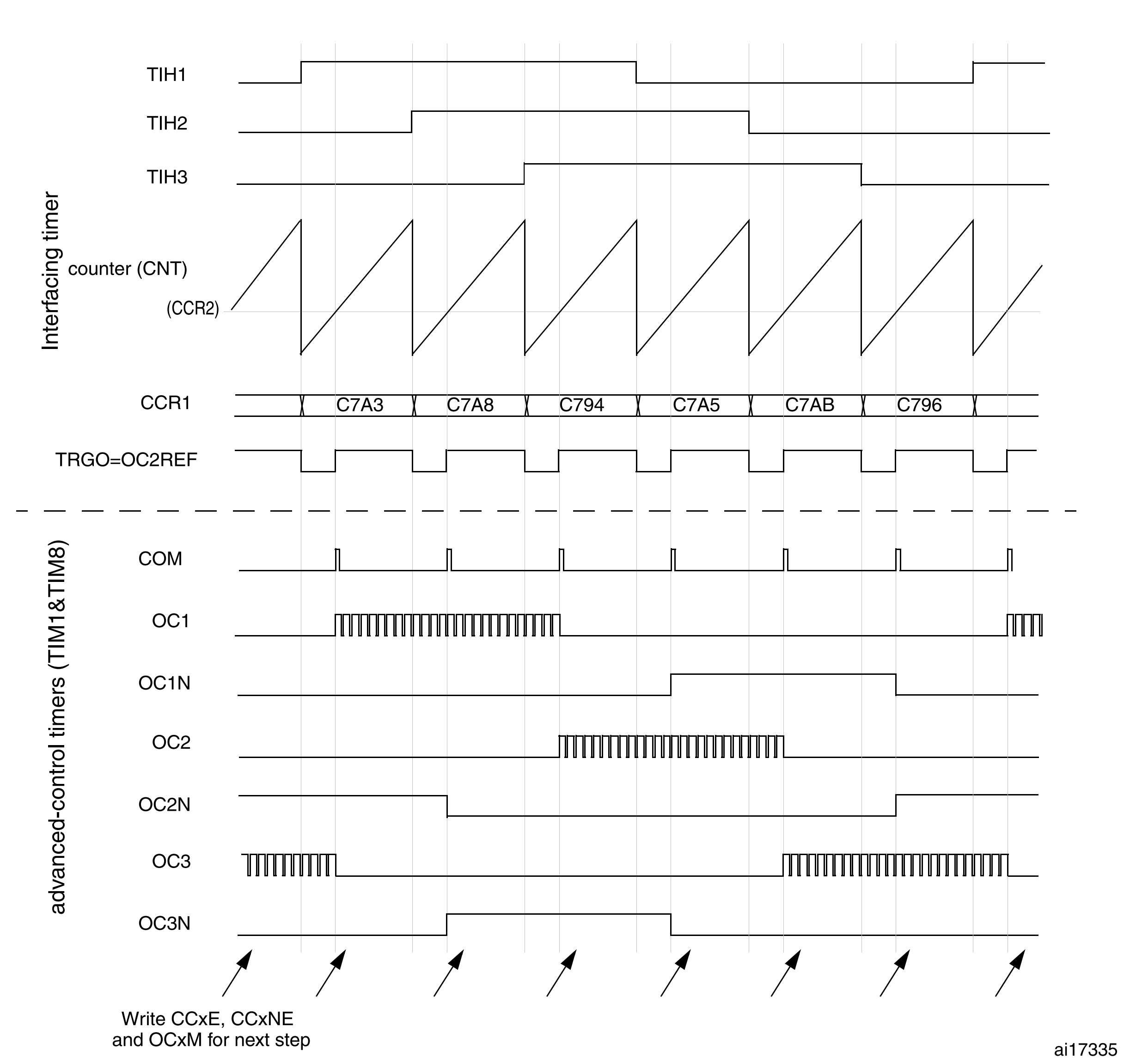
Example: one wants to change the PWM configuration of the advanced-control timer TIM1 after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the TIMx_CR2 register to ‘1’,
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program the channel 1 in capture mode (TRC selected): write the CC1S bits in the TIMx_CCMR1 register to ‘11’. The digital filter can also be programmed if needed,
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the TIMx_CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the TIMx_CR2 register to ‘101’,

In the advanced-control timer TIM1, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

The *Figure 367* describes this example.

Figure 367. Example of Hall sensor interface



38.3.26 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 39.3.19: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

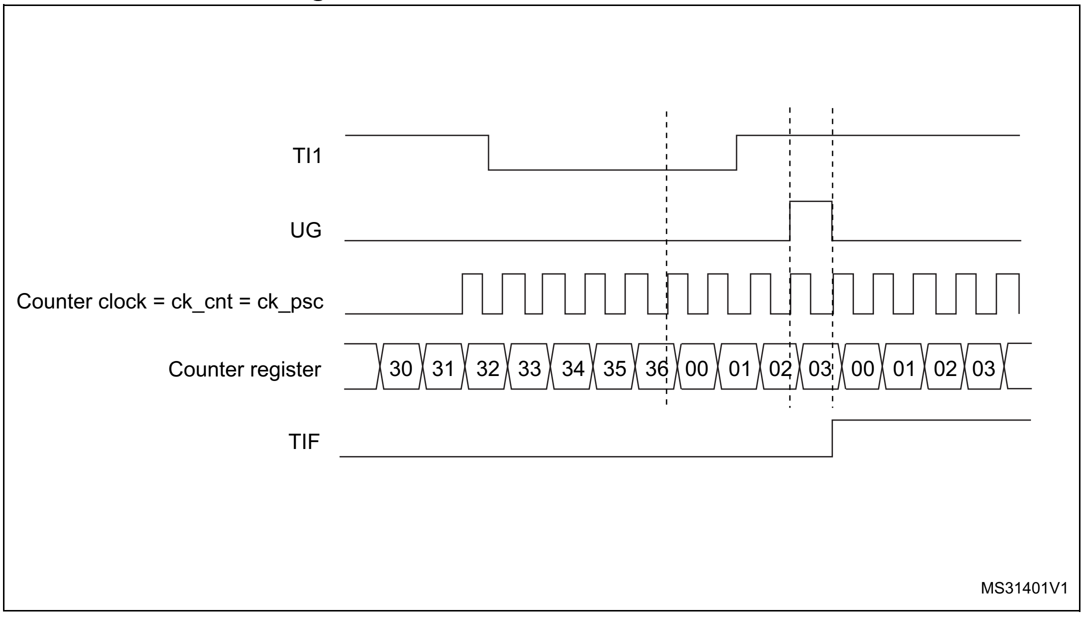
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 368. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

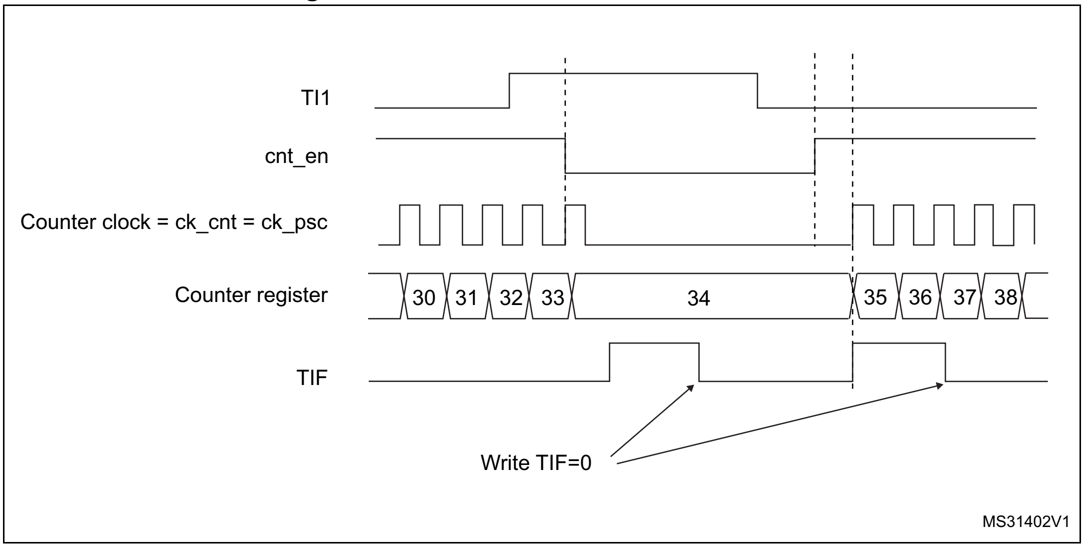
In the following example, the upcounter counts only when TI1 input is low:

- Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
- Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 369. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

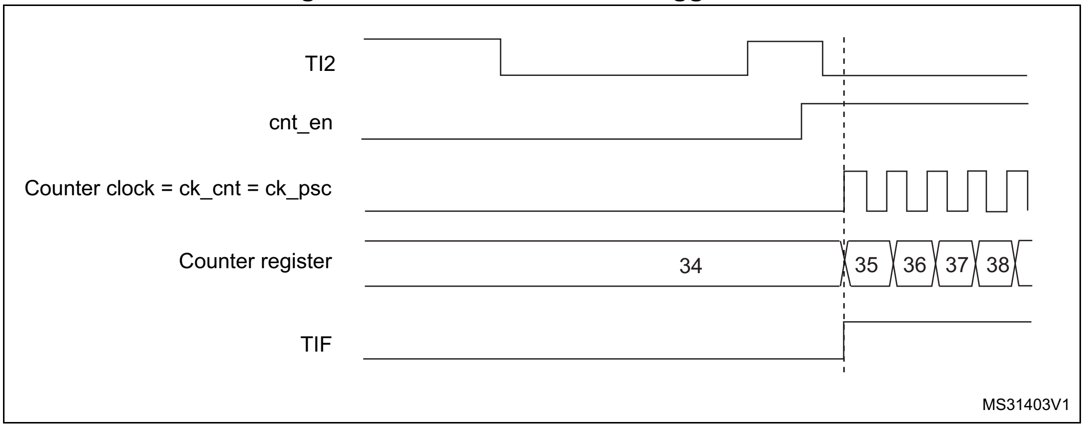
- Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

- Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 370. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

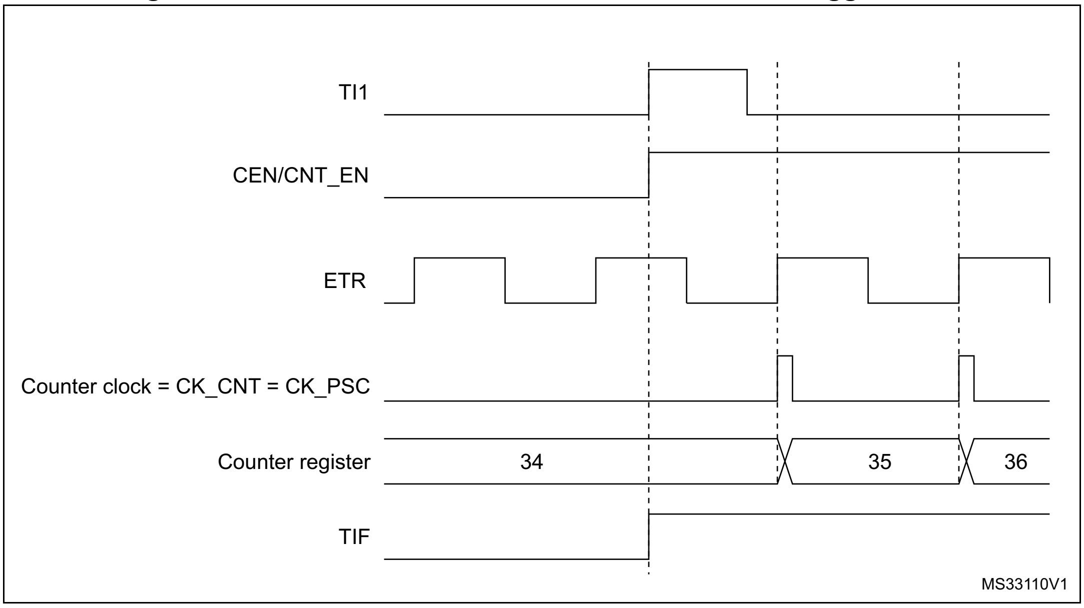
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   – ETF = 0000: no filter
   – ETPS = 00: prescaler disabled
   – ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   – IC1F = 0000: no filter.
   – The capture prescaler is not used for triggering and does not need to be configured.
   – CC1S = 01 in TIMx_CCMR1 register to select only the input capture source
   – CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 371. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.3.27 ADC synchronization

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the TRGO2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 351 on page 1935*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Note: The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

38.3.28 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.3.29 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBG module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

38.4 TIM1/TIM8 registers

Refer to for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.4.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx):

00: $t_{DTS}=t_{CK\_INT}$
01: $t_{DTS}=2*t_{CK\_INT}$
10: $t_{DTS}=4*t_{CK\_INT}$
11: Reserved, do not program this value

Note: $t_{DTS} = 1/f_{DTS}$, $t_{CK\_INT} = 1/f_{CK\_INT}$.

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: Switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1) is not allowed

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.4.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
								rw	rw	rw	rw		rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (TRGO2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on TRGO2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (TRGO2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (TRGO2).

0100: **Compare** - OC1REFC signal is used as trigger output (TRGO2)

0101: **Compare** - OC2REFC signal is used as trigger output (TRGO2)

0110: **Compare** - OC3REFC signal is used as trigger output (TRGO2)

0111: **Compare** - OC4REFC signal is used as trigger output (TRGO2)

1000: **Compare** - OC5REFC signal is used as trigger output (TRGO2)

1001: **Compare** - OC6REFC signal is used as trigger output (TRGO2)

1010: **Compare Pulse** - OC4REFC rising or falling edges generate pulses on TRGO2

1011: **Compare Pulse** - OC6REFC rising or falling edges generate pulses on TRGO2

1100: **Compare Pulse** - OC4REFC or OC6REFC rising edges generate pulses on TRGO2

1101: **Compare Pulse** - OC4REFC rising or OC6REFC falling edges generate pulses on TRGO2

1110: **Compare Pulse** - OC5REFC or OC6REFC rising edges generate pulses on TRGO2

1111: **Compare Pulse** - OC5REFC rising or OC6REFC falling edges generate pulses on TRGO2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output Idle state 6 (OC6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output Idle state 5 (OC5 output)

Refer to OIS1 bit

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS4**: Output Idle state 4 (OC4 output)

Refer to OIS1 bit

Bit 13 **OIS3N**: Output Idle state 3 (OC3N output)

Refer to OIS1N bit

Bit 12 **OIS3**: Output Idle state 3 (OC3 output)

Refer to OIS1 bit

Bit 11 **OIS2N**: Output Idle state 2 (OC2N output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output Idle state 2 (OC2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)
0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)
0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: TI1 selection
0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow selected information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

38.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge.

1: ETR is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of f_{CK_INT} frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF
01: ETRP frequency divided by 2
10: ETRP frequency divided by 4
11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
00111: External Trigger input (ETRF)
Others: Reserved

See *Table 278: TIMx internal trigger connection on page 1967* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = ‘1’ then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.

0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Codes above 1000: Reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 278. TIMx internal trigger connection

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM1	TIM15	TIM2	TIM3	TIM4
TIM8	TIM1	TIM2	TIM4	TIM5

38.4.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.4.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF
														rc_w0	rc_w0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag

Refer to CC1IF description (Note: Channel 6 can only be configured as output)

Bit 16 **CC5IF**: Compare 5 interrupt flag

Refer to CC1IF description (Note: Channel 5 can only be configured as output)

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System Break interrupt flag

This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.

This flag must be reset to re-start PWM operation.

0: No break event occurred.

1: An active level has been detected on the system break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag

Refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag

Refer to CC1OF description

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag

Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE=1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

Refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 38.4.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

38.4.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: When CCPC bit is set, it allows CCxE, CCxNE and OCxM bits to be updated.

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/Compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/Compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. The prescaler internal counter is also cleared (the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

38.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Refer to IC1PSC[1:0] description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI1

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

38.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output Compare 2 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode
Refer to OC1M[3:0] description.

Bit 11 **OC2PE**: Output Compare 2 preload enable
Refer to OC1PE description.

Bit 10 **OC2FE**: Output Compare 2 fast enable
Refer to OC1FE description.

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 **OC1CE:** Output Compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0’) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=’1’).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S=’00’ (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: On channels having a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI2

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

38.4.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Refer to IC1F[3:0] description.

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler
Refer to IC1PSC[1:0] description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter
Refer to IC1F[3:0] description.

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler
Refer to IC1PSC[1:0] description.

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

38.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (e.g. channel 1 in input capture mode and channel 2 in output compare mode).

Output compare mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable
Refer to OC1CE description.

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] description.

Bit 11 **OC4PE**: Output compare 4 preload enable
Refer to OC1PE description.

Bit 10 **OC4FE**: Output compare 4 fast enable
Refer to OC1FE description.

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = '0' in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable
Refer to OC1CE description.

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M[3:0] description.

Bit 3 **OC3PE**: Output compare 3 preload enable
Refer to OC1PE description.

Bit 2 **OC3FE**: Output compare 3 fast enable
Refer to OC1FE description.

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in TIMx_CCER).

38.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/Compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/Compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/Compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/Compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/Compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/Compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/Compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/Compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high.

1: OC1N active low.

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (channel configured as output).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: The configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 279* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 279. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states[(1)]	
MOE bit	**OSSI bit**	**OSSR bit**	**CCxE bit**	**CCxNE bit**	**OCx output state**	**OCxN output state**
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) OCx=0, OCxN=0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN = OCxREF xor CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN = OCxREF x or CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP	Off-State (output enabled with inactive state) OCxN=CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP (if BRK or BRK2 is triggered). Then (this is valid only if BRK is triggered), if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state (may cause a short circuit when driving switches in half-bridge configuration). **Note**: BRK2 can only be used if OSSI = OSSR = 1.	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO registers.

38.4.12 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

38.4.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

38.4.14 TIMx auto-reload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 38.3.1: Time-base unit on page 1903* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

38.4.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

the number of PWM periods in edge-aligned mode

the number of half PWM period in center-aligned mode.

38.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

38.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC2 output.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

38.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR3[15:0]**: Capture/Compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.

If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

38.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR4[15:0]**: Capture/Compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

38.4.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BK2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional

Refer to BKBID description

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 Disarm

Refer to BKDSRM description

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity

0: Break input BRK2 is active low
1: Break input BRK2 is active high

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable

This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per *Figure 355: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled

Note: The BKIN2 must only be used with OSSR = OSSI = 1.

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bit-field defines the frequency used to sample BRK2 input and the length of the digital filter applied to BRK2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK2 acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (BRK or BRK2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details (*Section 38.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs BRK and BRK2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BRK sources, as per *Figure 355: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 38.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details (*Section 38.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and B*K2BID, BKBID, BK2DSRM, BKDSRM, BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0]* bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x t_{DTG} with $t_{DTG} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t_{DTG} with t_{DTG} = 2 x t_{DTS}.
DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{DTG} with t_{DTG} = 8 x t_{DTS}.
DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{DTG} with t_{DTG} = 16 x t_{DTS}.
Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.4.21 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
10001: 18 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIMx_CR1.

– If DBL = 7 bytes and DBA = TIMx_CR1 represents the address of the byte to be transferred, the address of the transfer should be given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data is copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

– If the DMA Data Size is configured in half-words, 16-bit data is transferred to each of the 7 registers.

– If the DMA Data Size is configured in bytes, the data is also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

38.4.22 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

38.4.23 TIMx capture/compare mode register 3 (TIMx_CCMR3)(x = 1, 8)

Address offset: 0x54

Reset value: 0x0000 0000

The channels 5 and 6 can only be configured in output.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable
Refer to OC1CE description.

Bits 24, 14, 13, 12 **OC6M[3:0]**: Output compare 6 mode
Refer to OC1M description.

Bit 11 **OC6PE**: Output compare 6 preload enable
Refer to OC1PE description.

Bit 10 **OC6FE**: Output compare 6 fast enable
Refer to OC1FE description.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable
Refer to OC1CE description.

Bits 16, 6, 5, 4 **OC5M[3:0]**: Output compare 5 mode
Refer to OC1M description.

Bit 3 **OC5PE**: Output compare 5 preload enable
Refer to OC1PE description.

Bit 2 **OC5FE**: Output compare 5 fast enable
Refer to OC1FE description.

Bits 1:0 Reserved, must be kept at reset value.

38.4.24 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group Channel 5 and Channel 3

Distortion on Channel 3 output:
0: No effect of OC5REF on OC3REFC
1: OC3REFC is the logical AND of OC3REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group Channel 5 and Channel 2

Distortion on Channel 2 output:
0: No effect of OC5REF on OC2REFC
1: OC2REFC is the logical AND of OC2REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group Channel 5 and Channel 1

Distortion on Channel 1 output:
0: No effect of OC5REF on OC1REFC5
1: OC1REFC is the logical AND of OC1REFC and OC5REF
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).

Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR5[15:0]**: Capture/Compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC5 output.

38.4.25 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x5C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR6[15:0]**: Capture/Compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC6 output.

38.4.26 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRSEL[1:0]		Res.	Res.	Res.	Res.	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
rw	rw					rw	rw								rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: ETR input is connected to I/O
0001: Reserved
0010: Reserved
0011: ADC1 AWD1
0100: ADC1 AWD2
0101: ADC1 AWD3
0110: ADC2 AWD1
0111: ADC2 AWD2
1000: ADC2 AWD3
Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable

This bit enables the dfsdm1_break[0] for the timer's BRK input. dfsdm1_break[0] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[0] input disabled

1: dfsdm1_break[0] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 334: TIM1/TIM8 ETR input circuitry and to Figure 355: Break and Break2 circuitry overview.

38.4.27 TIM1 Alternate function register 2 (TIM1_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	BK2 INP	BK2DF1 BK1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2INE
						rw	rw								rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK1E**: BRK2 dfsdm1_break[1] enable

This bit enables the dfsdm1_break[1] for the timer's BRK2 input. dfsdm1_break[1] output is 'ORed' with the other BRK2 sources.

0: dfsdm1_break[1] input disabled

1: dfsdm1_break[1] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 355: Break and Break2 circuitry overview.

38.4.28 TIM8 Alternate function option register 1 (TIM8_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
rw	rw					rw	rw								rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.

0000: ETR input is connected to I/O

0001: Reserved

0010: Reserved

0011: ADC1 AWD1

0100: ADC1 AWD2

0101: ADC1 AWD3

0110: ADC2 AWD1

0111: ADC2 AWD2

1000: ADC2 AWD3

Others: Reserved

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input polarity is not inverted (active low if BKP=0, active high if BKP=1)

1: BKIN input polarity is inverted (active high if BKP=0, active low if BKP=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable

This bit enables the dfsdm1_break[2] for the timer's BRK input. dfsdm1_break[2] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[2] input disabled

1: dfsdm1_break[2] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 334: TIM1/TIM8 ETR input circuitry and to Figure 355: Break and Break2 circuitry overview.

38.4.29 TIM8 Alternate function option register 2 (TIM8_AF2)

Address offset: 0x64

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	BK2 INP	BK2DF1 BK3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2INE
						rw	rw								rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BK2INP**: BRK2 BKIN2 input polarity

This bit selects the BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: BKIN2 input polarity is not inverted (active low if BK2P=0, active high if BK2P=1)

1: BKIN2 input polarity is inverted (active high if BK2P=0, active low if BK2P=1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2DF1BK3E**: BRK2 dfsdm1_break[3] enable

This bit enables the dfsdm1_break[3] for the timer's BRK2 input. dfsdm1_break[3] output is 'ORed' with the other BRK2 sources.

0: dfsdm1_break[3] input disabled

1: dfsdm1_break[3] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BK2INE**: BRK2 BKIN input enable

This bit enables the BKIN2 alternate function input for the timer's BRK2 input. BKIN2 input is 'ORed' with the other BRK2 sources.

0: BKIN2 input disabled

1: BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Figure 355: Break and Break2 circuitry overview.

38.4.30 TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input

0000: TIM1_CH4 input

Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input

0000: TIM1_CH3 input

Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM1_CH2 input

Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM1_CH1 input

Others: Reserved

38.4.31 TIM8 timer input selection register (TIM8_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: selects TI4[0] to TI4[15] input
0000: TIM8_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: selects TI3[0] to TI3[15] input
0000: TIM8_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIM8_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM8_CH1 input
Others: Reserved

38.4.32 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

Table 280. TIM1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM1_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMAP	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	TIM1_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	TIM1_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIM1_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	TIM1_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	TIM1_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM1_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM1_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM1_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIM1_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 280. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	TIM1_CNT	UIFCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM1_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM1_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM1_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	TIM1_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM1_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIM1_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIM1_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIM1_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIM1_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIM1_DMAR	DMAB[31:0]																															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	TIM1_CCMR3 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	TIM1_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 280. TIM1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C	TIM1_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIM1_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				Res.	Res.	Res.	Res.	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
	Reset value															0	0	0	0					0	0								1
0x64	TIM1_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2INP	BK2DF1BK1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2INE
	Reset value																							0	0								1
0x68	TIM1_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

38.4.33 TIM8 register map

TIM8 registers are mapped as 16-bit addressable registers as described in the table below:

Table 281. TIM8 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM8_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	TIM8_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value									0	0	0	0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	TIM8_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	TIM8_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	TIM8_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value															0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 281. TIM8 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	**TIM8_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0
0x18	**TIM8_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM8_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIM8_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM8_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM8_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	Res.	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIM8_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM8_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM8_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM8_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM8_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**TIM8_CCR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**TIM8_CCR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 281. TIM8 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x44	TIM8_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIM8_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIM8_DMAR	DMAB[31:0]																															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	TIM8_CCMR3 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x58	TIM8_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[15:0]															
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	TIM8_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIM8_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				Res.	Res.	Res.	Res.	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
	Reset value															0	0	0	0					0	0								1
0x64	TIM8_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2INP	BK2DF1BK3E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BK2INE
	Reset value																							0	0								1
0x68	TIM8_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

39 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

39.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit/32-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

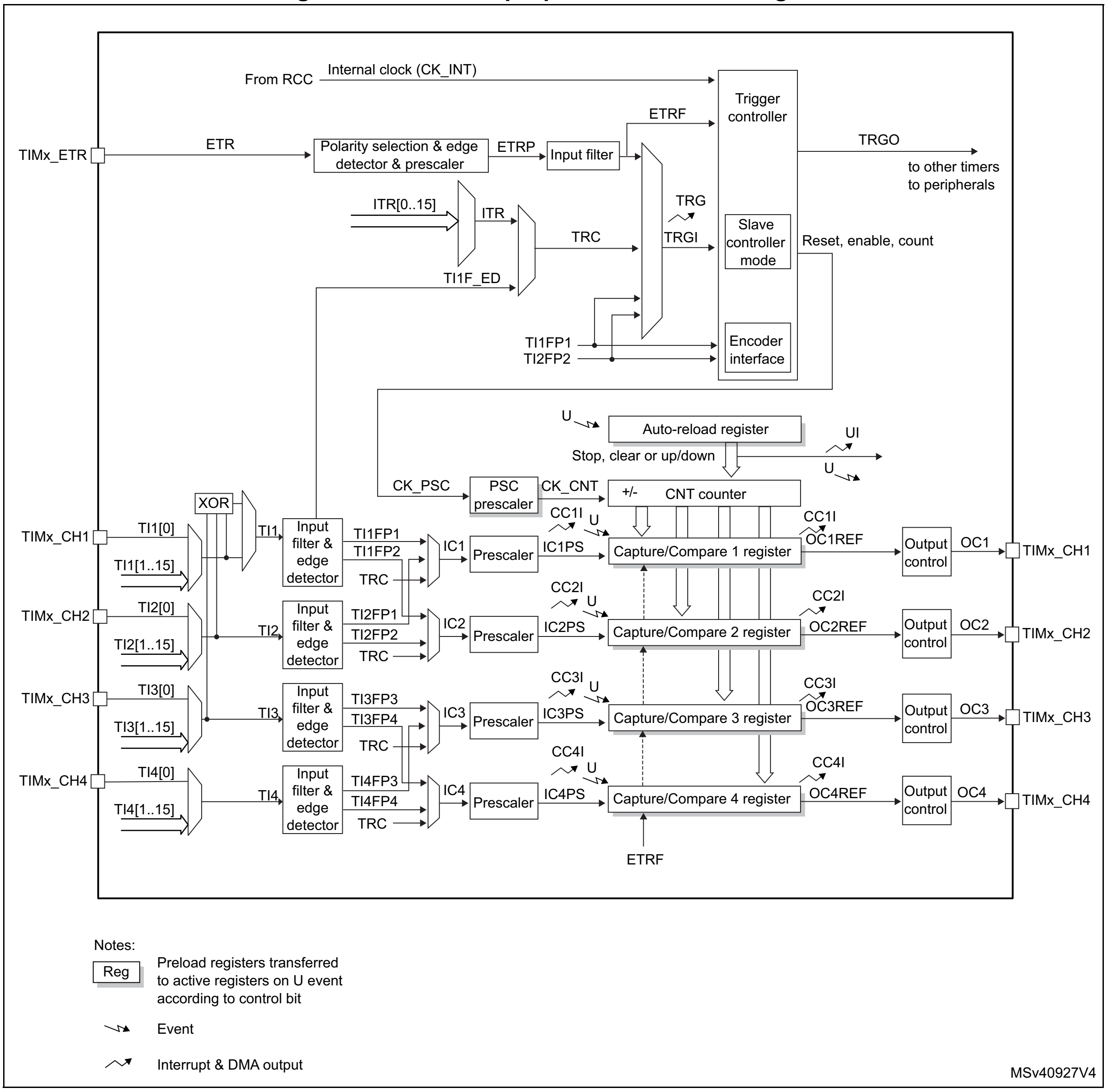
The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 39.3.19: Timer synchronization*.

39.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit (TIM3, TIM4) or 32-bit (TIM2 and TIM5) up, down, up/down auto-reload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge- and Center-aligned modes)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 372. General-purpose timer block diagram



39.3 TIM2/TIM3/TIM4/TIM5 functional description

39.3.1 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related auto-reload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 373 and *Figure 374* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 373. Counter timing diagram with prescaler division change from 1 to 2

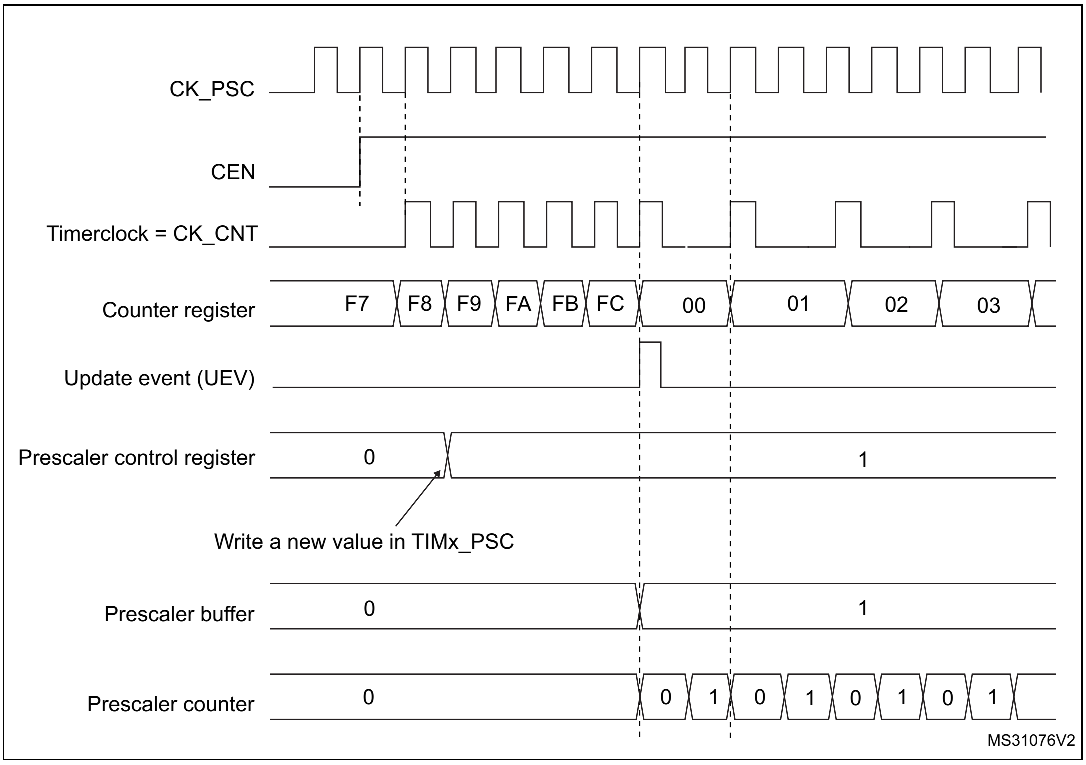
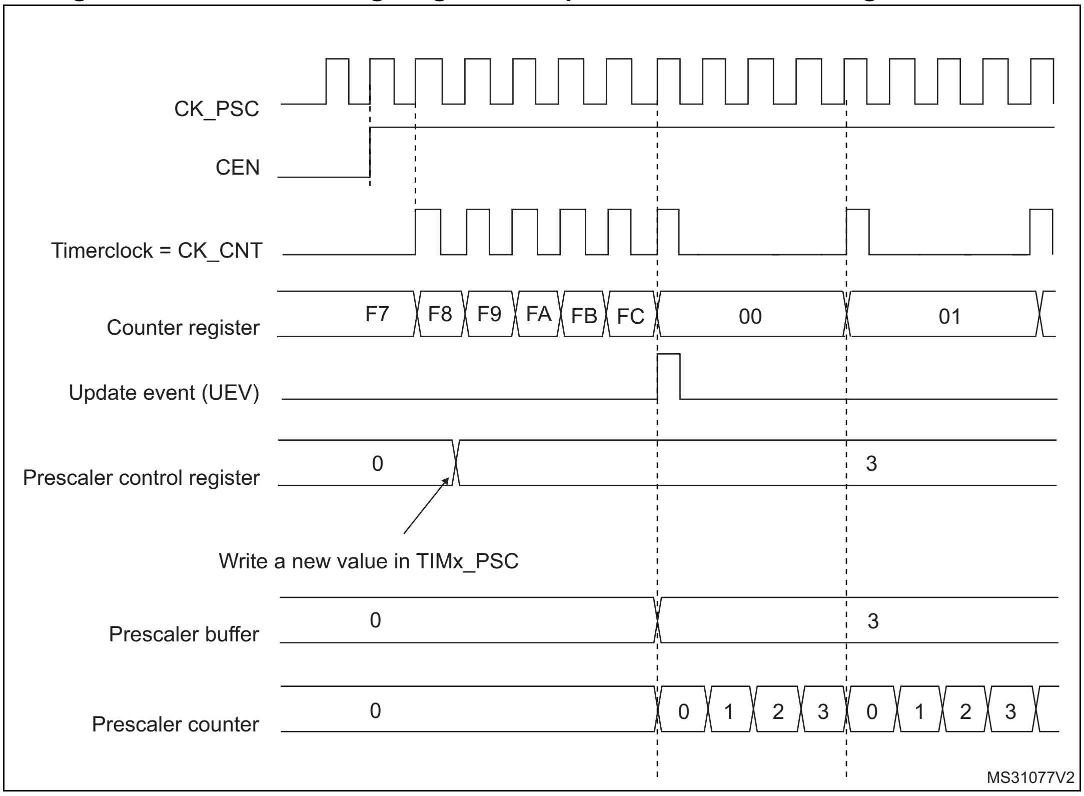


Figure 374. Counter timing diagram with prescaler division change from 1 to 4



39.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An Update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 375. Counter timing diagram, internal clock divided by 1

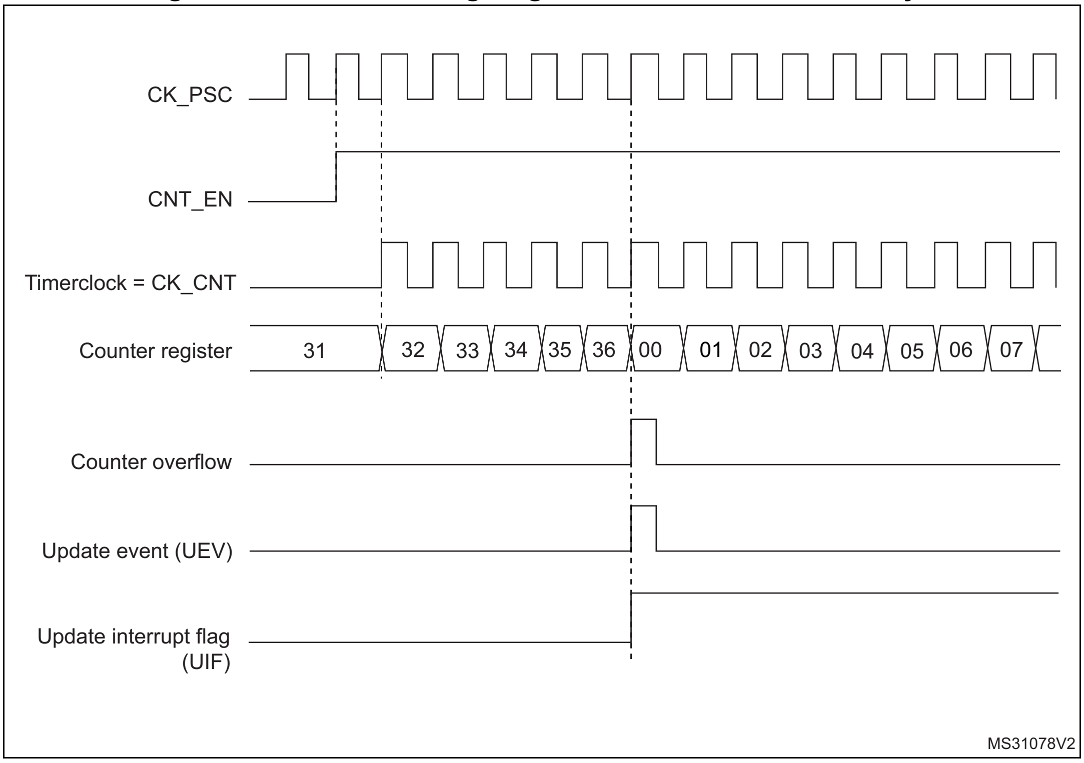


Figure 376. Counter timing diagram, internal clock divided by 2

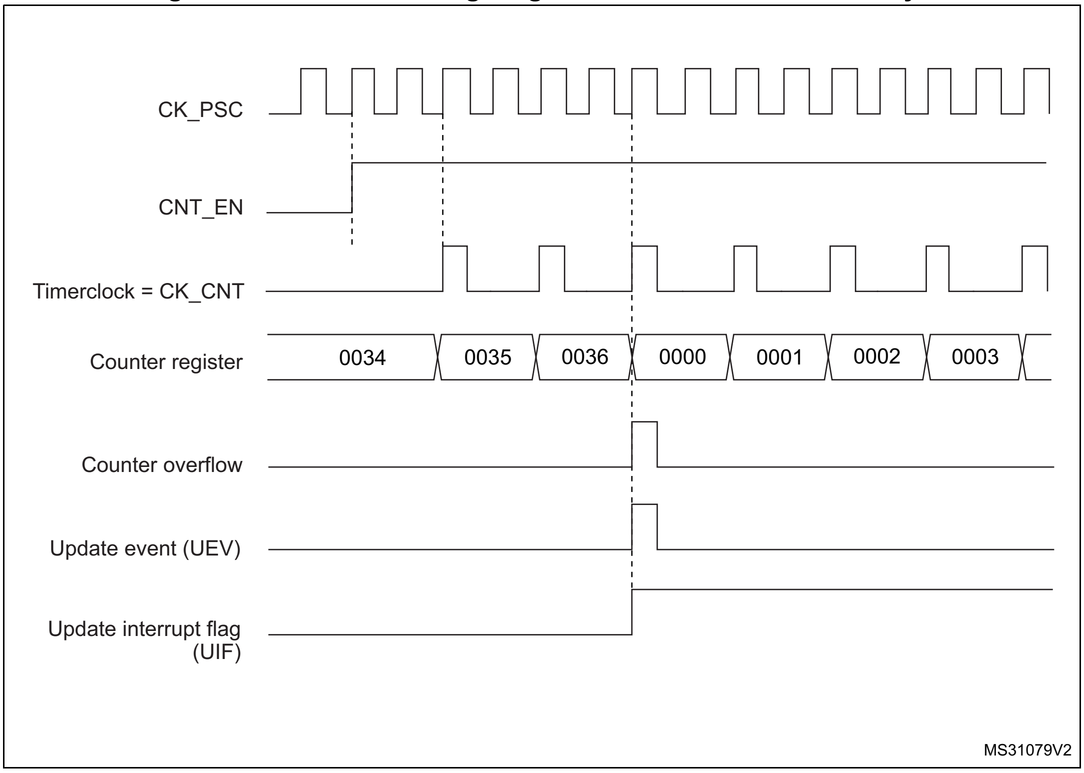


Figure 377. Counter timing diagram, internal clock divided by 4

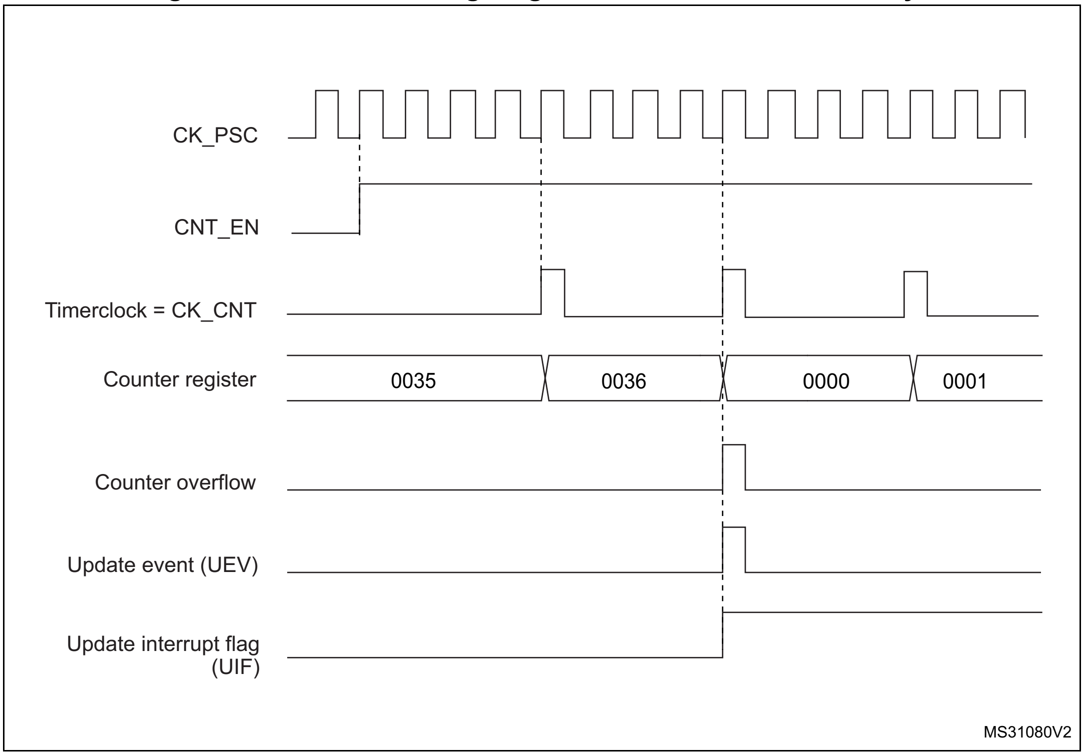


Figure 378. Counter timing diagram, internal clock divided by N

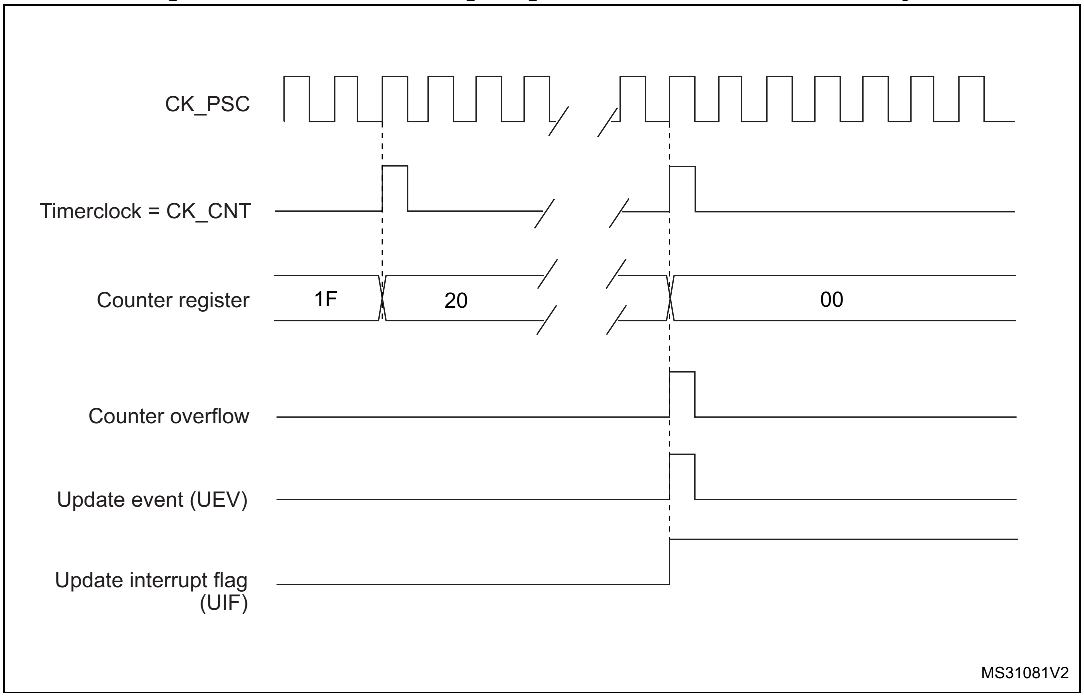


Figure 379. Counter timing diagram, Update event when ARPE=0 (TIMx_ARR not preloaded)

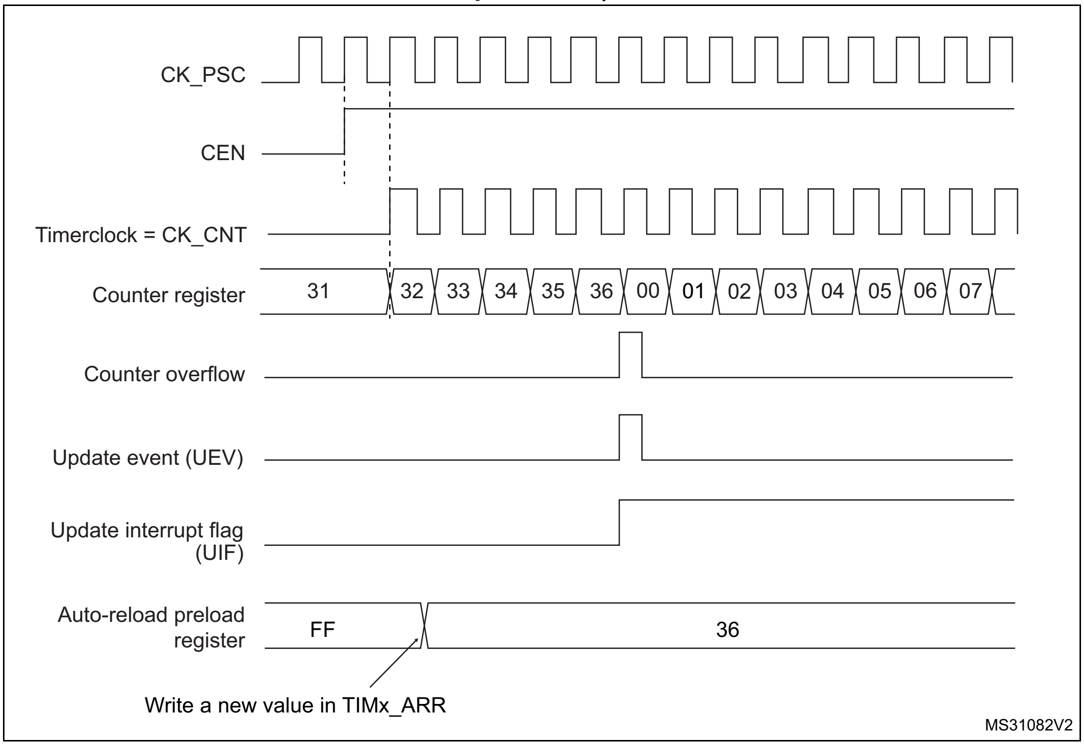
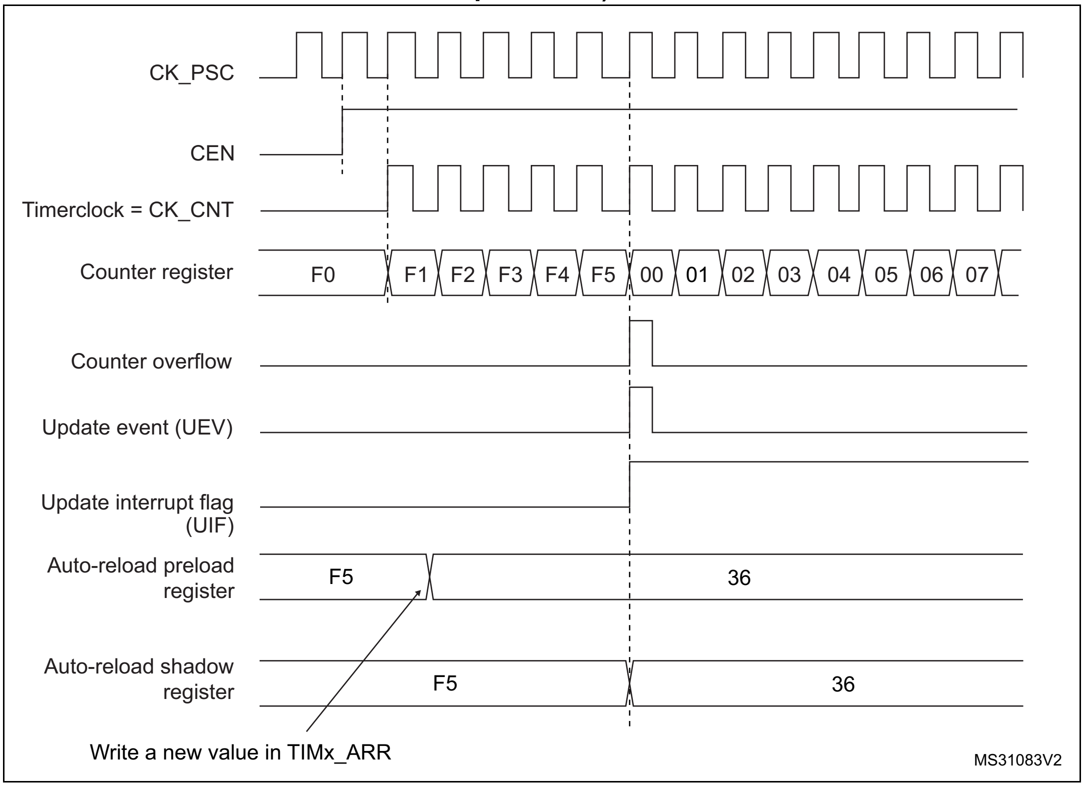


Figure 380. Counter timing diagram, Update event when ARPE=1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the auto-reload value (content of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An Update event can be generate at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 381. Counter timing diagram, internal clock divided by 1

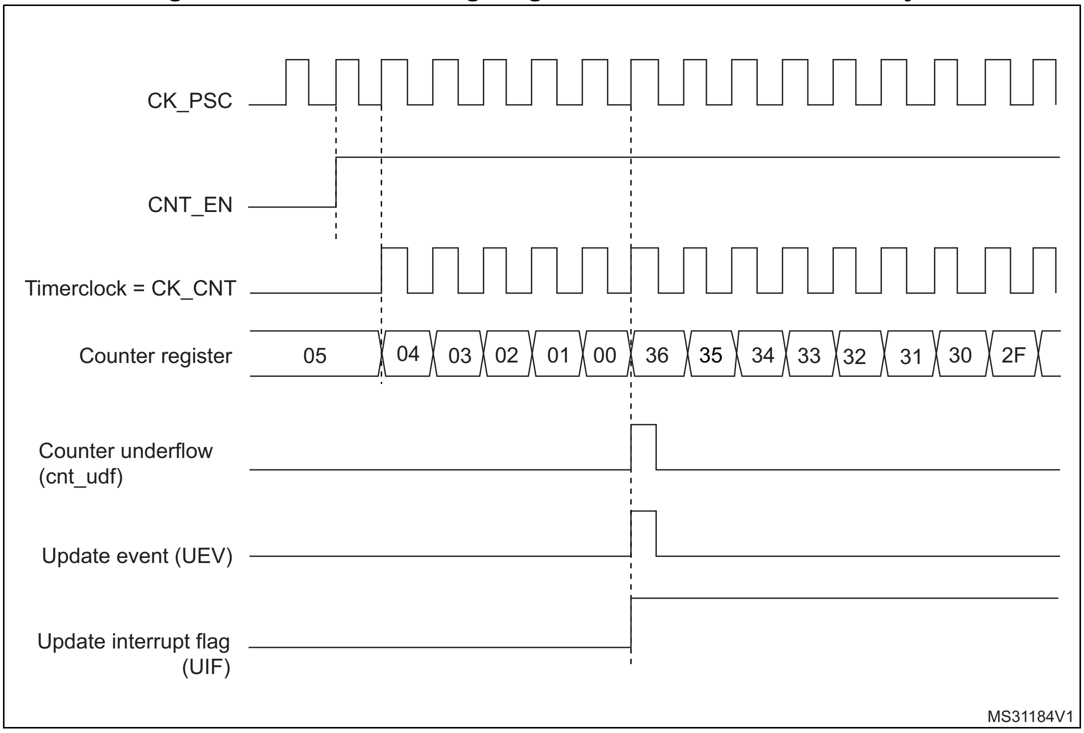


Figure 382. Counter timing diagram, internal clock divided by 2

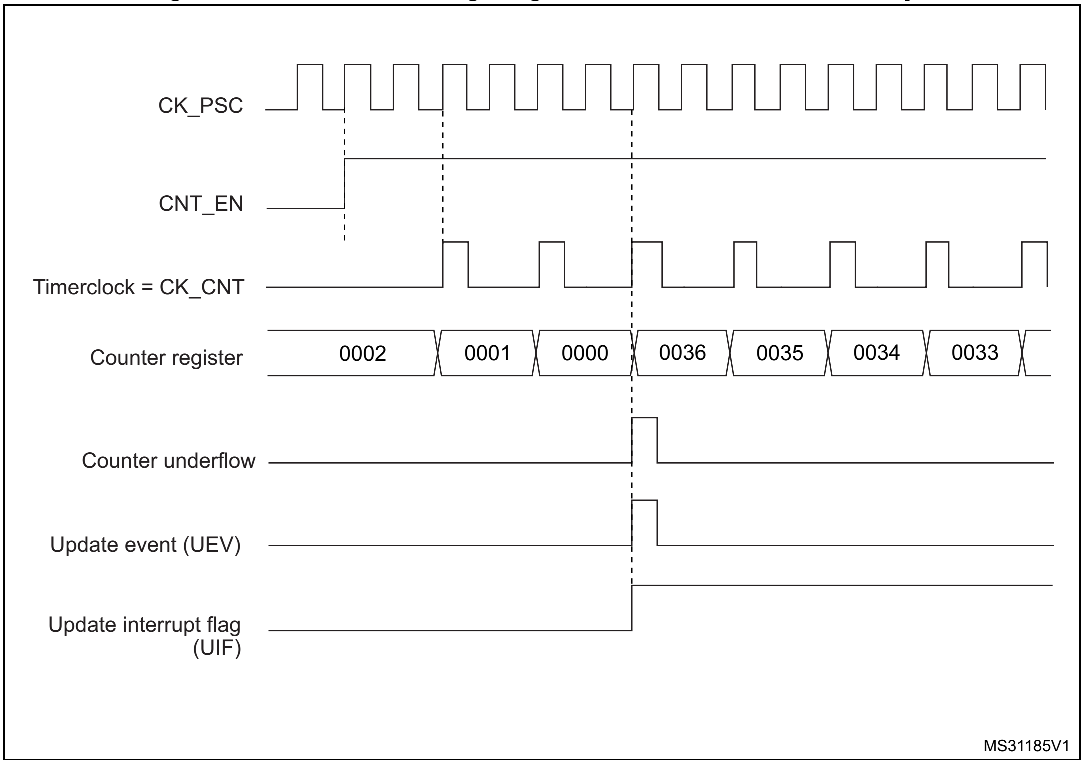


Figure 383. Counter timing diagram, internal clock divided by 4

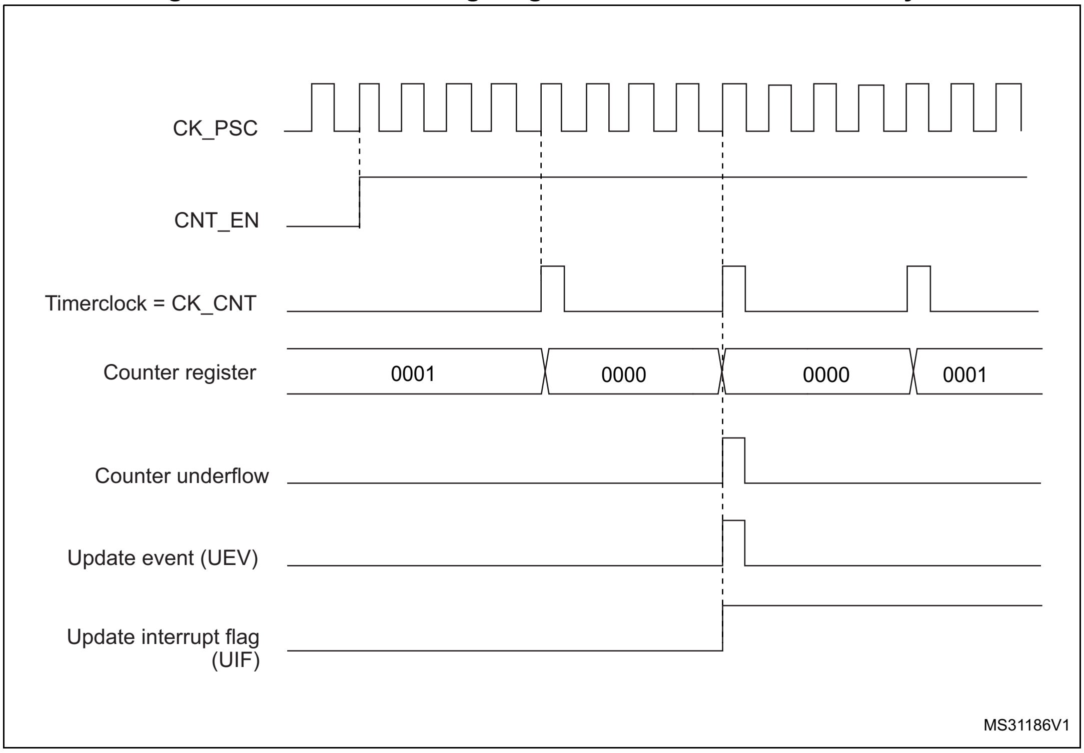


Figure 384. Counter timing diagram, internal clock divided by N

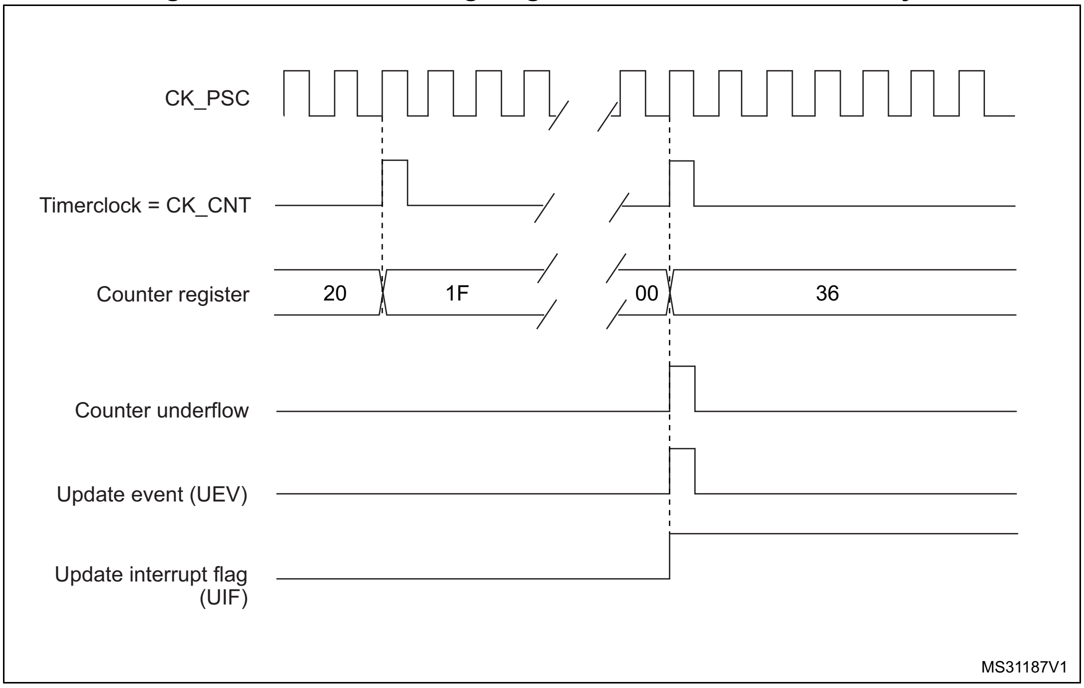
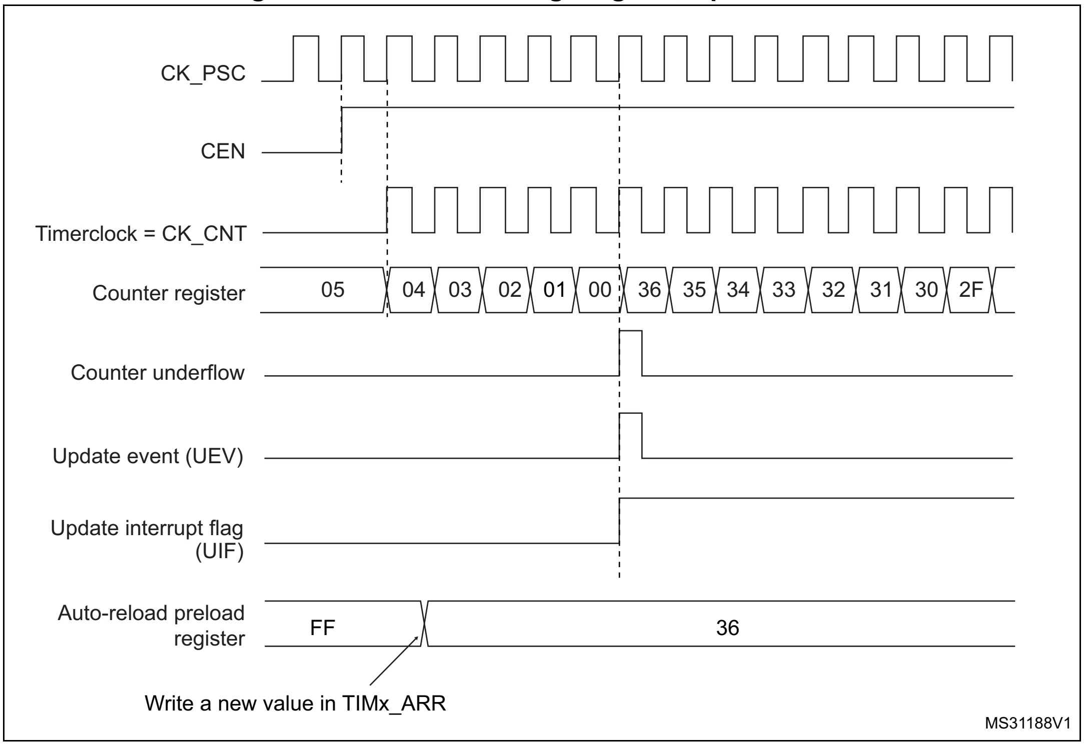


Figure 385. Counter timing diagram, Update event



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the auto-reload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or

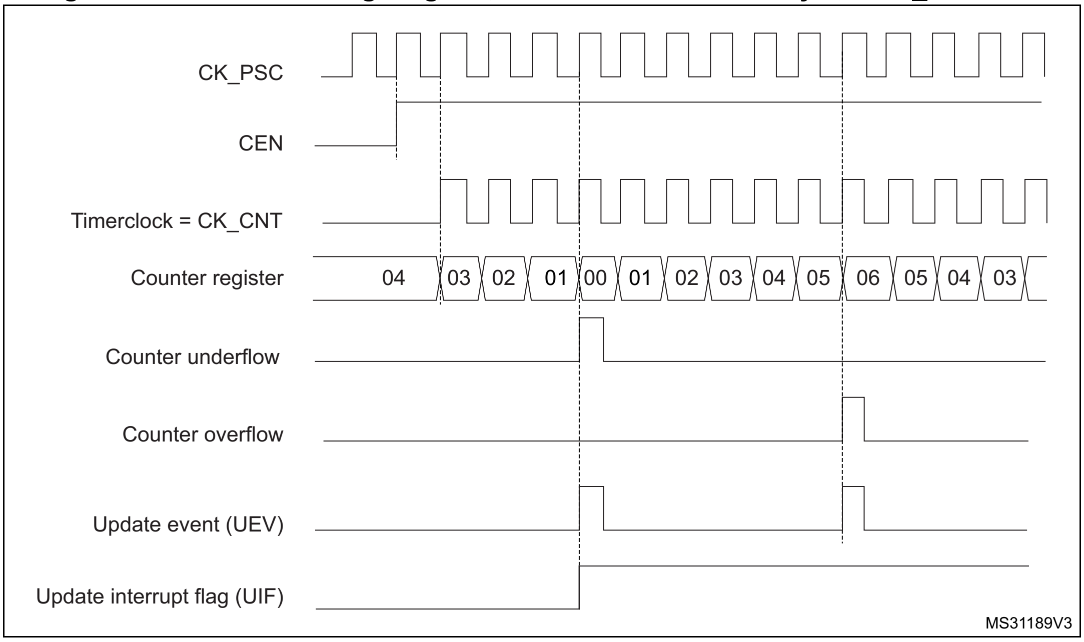
DMA request is sent). This is to avoid generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The auto-reload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 386. Counter timing diagram, internal clock divided by 1, TIMx_ARR=0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 39.4.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5) on page 2049*).

Figure 387. Counter timing diagram, internal clock divided by 2

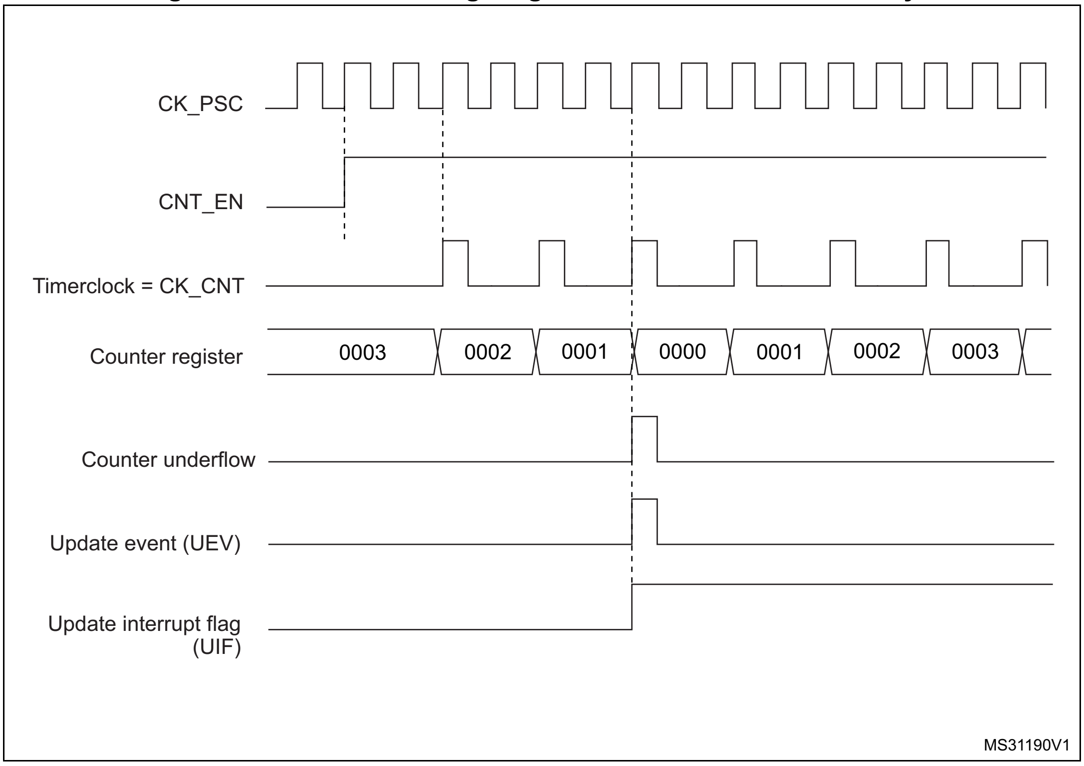
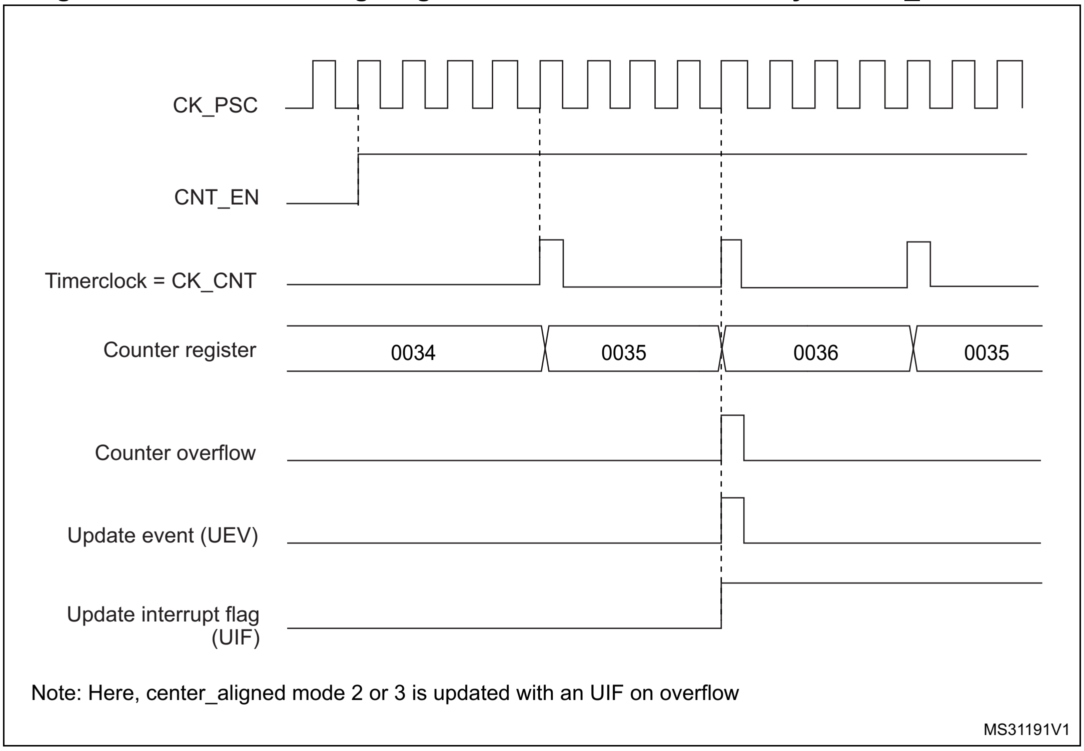


Figure 388. Counter timing diagram, internal clock divided by 4, TIMx_ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

Figure 389. Counter timing diagram, internal clock divided by N

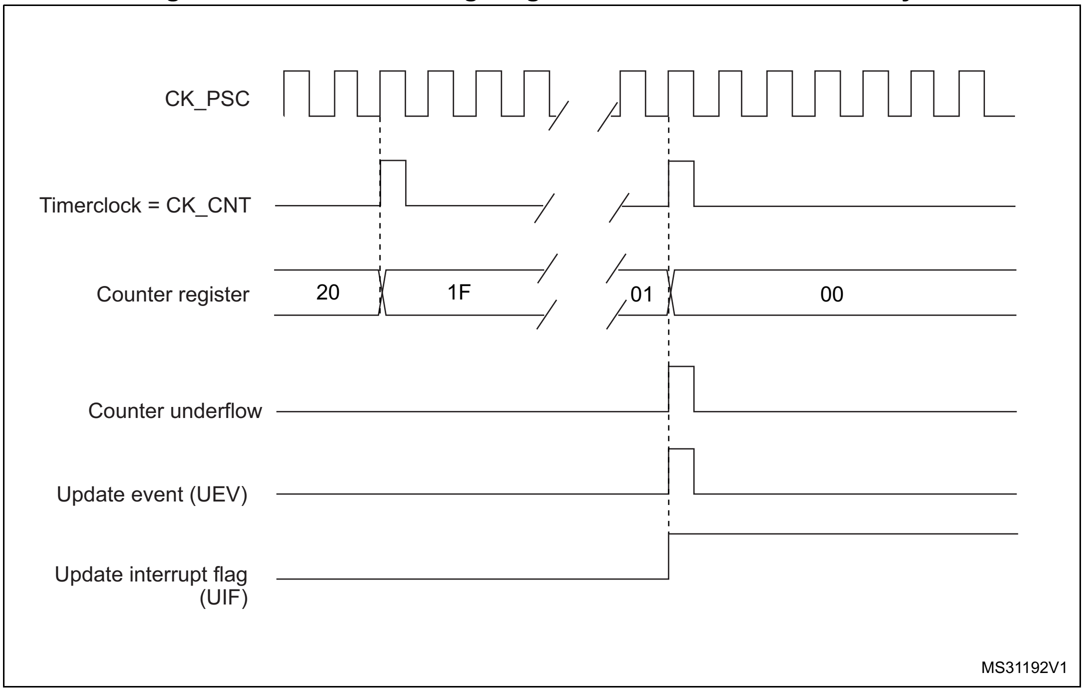


Figure 390. Counter timing diagram, Update event with ARPE=1 (counter underflow)

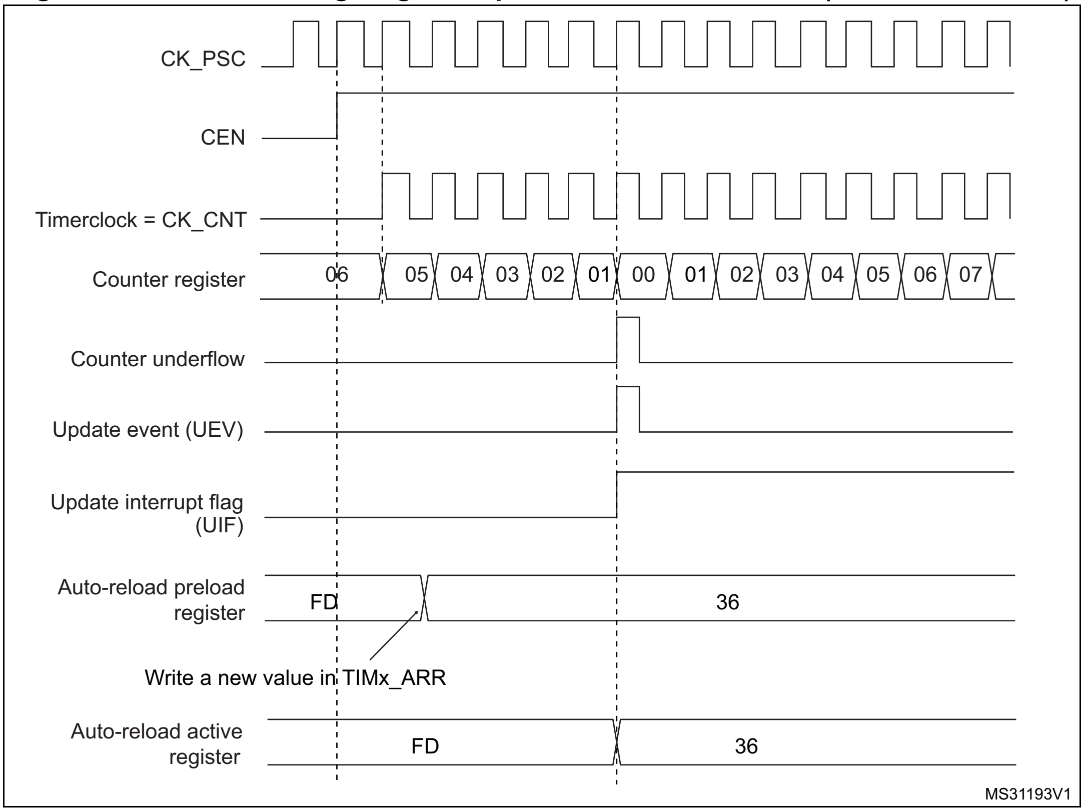
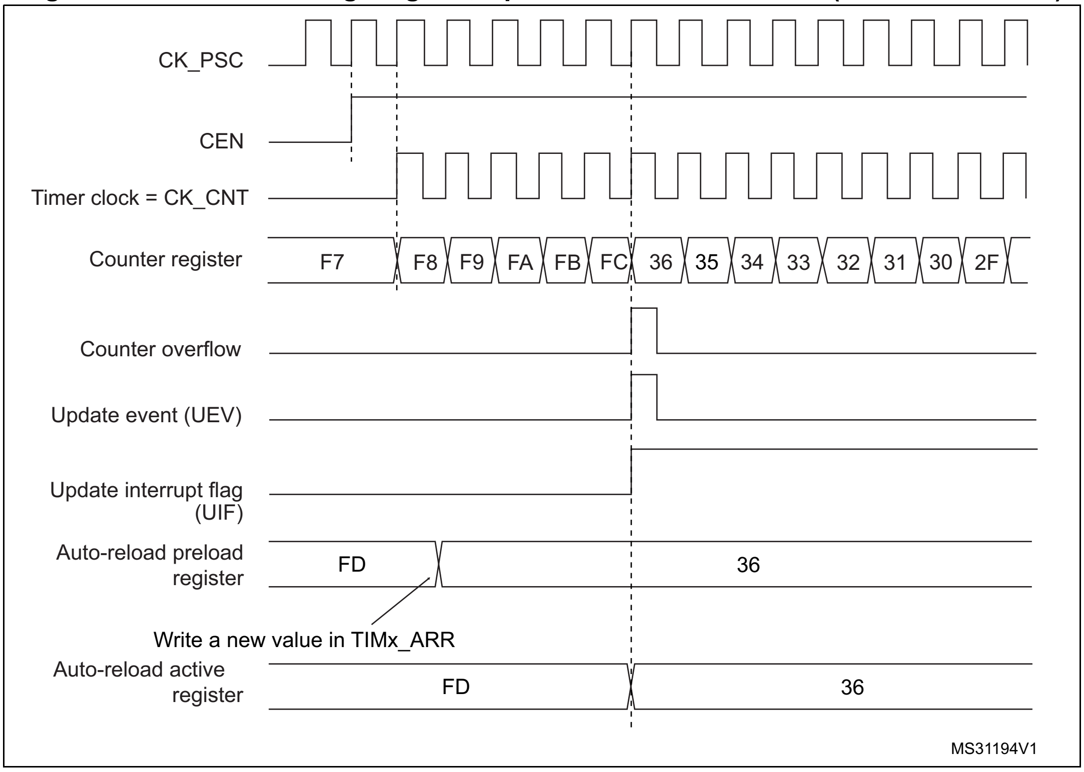


Figure 391. Counter timing diagram, Update event with ARPE=1 (counter overflow)



39.3.3 Clock selection

The counter clock can be provided by the following clock sources:

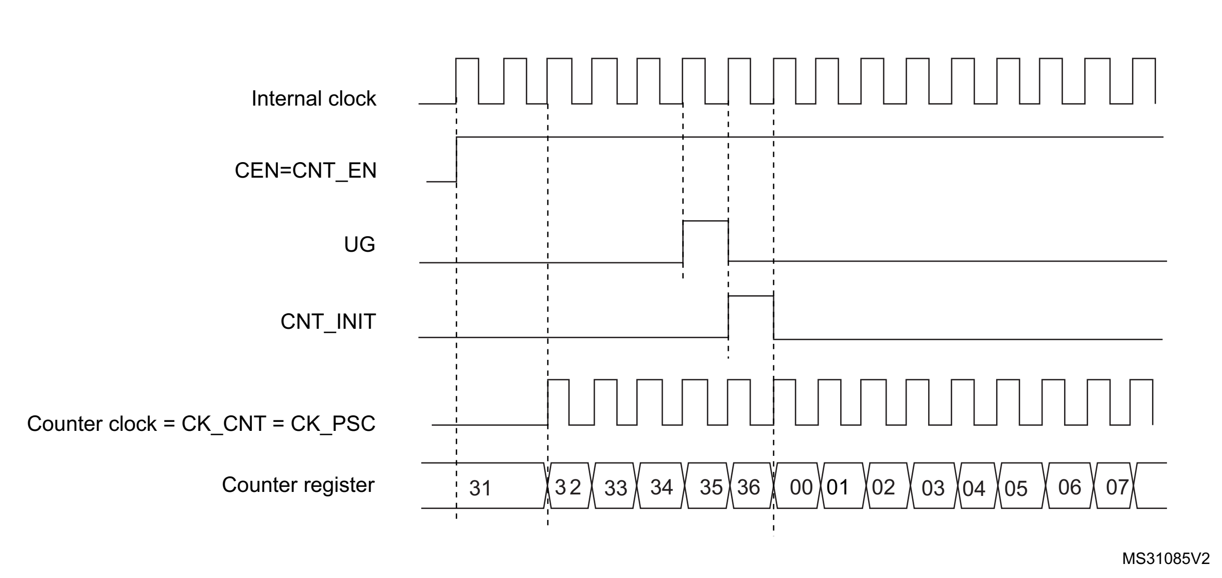
- Internal clock (CK_INT)
- External clock mode1: external input pin (TIx)
- External clock mode2: external trigger input (ETR)
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, Timer X can be configured to act as a prescaler for Timer Y. Refer to *: Using one timer as prescaler for another timer on page 2043* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 392 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

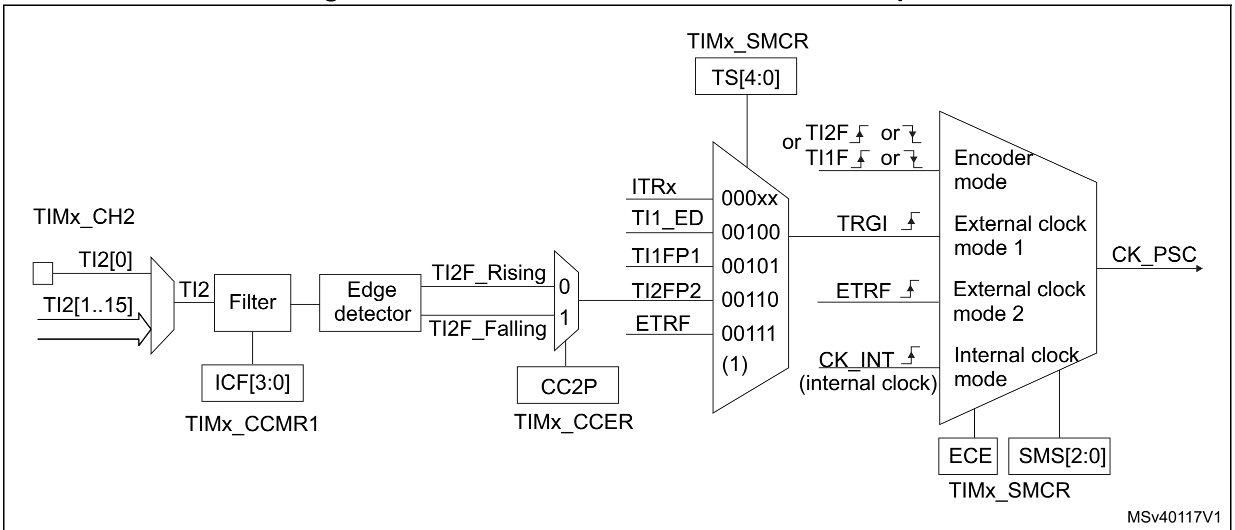
Figure 392. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 393. TI2 external clock connection example



1. Codes ranging from 01000 to 11111: ITRy.

For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S= ‘01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).

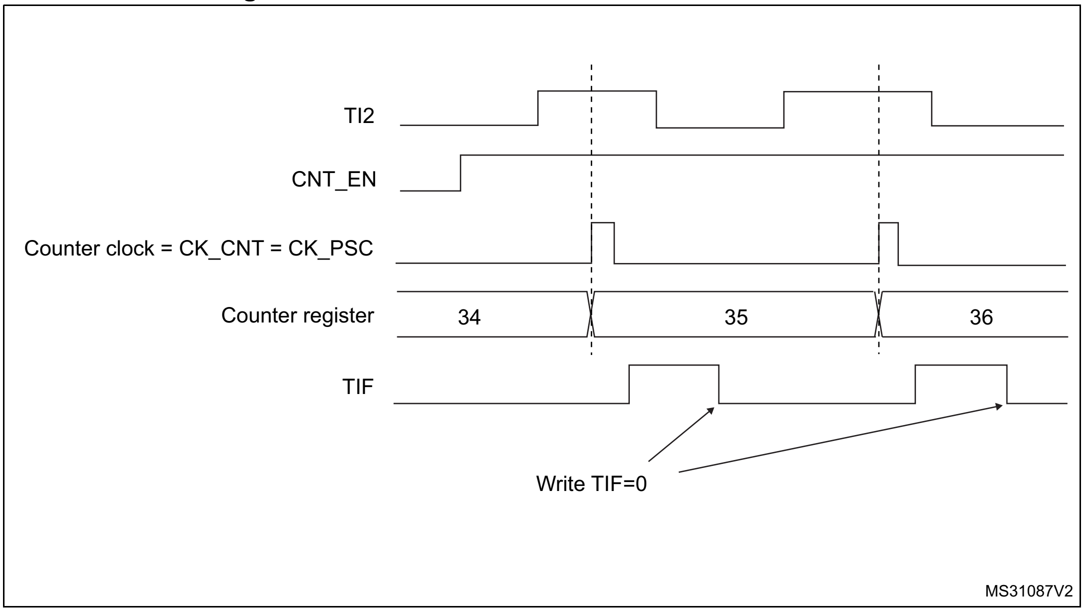
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P=0 and CC2NP=0 and CC2NP=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 394. Control circuit in external clock mode 1



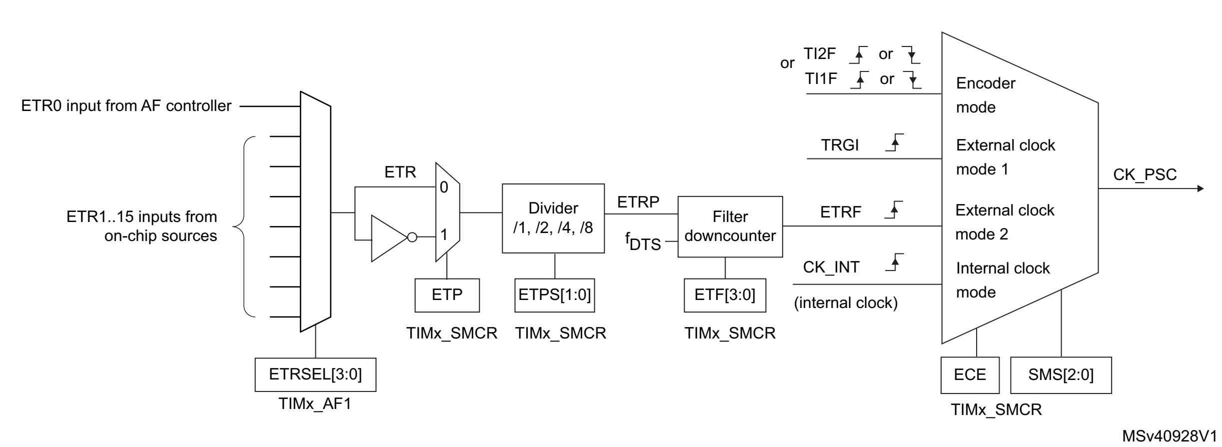
External clock source mode 2

This mode is selected by writing ECE=1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 395 gives an overview of the external trigger input block.

Figure 395. External trigger input block



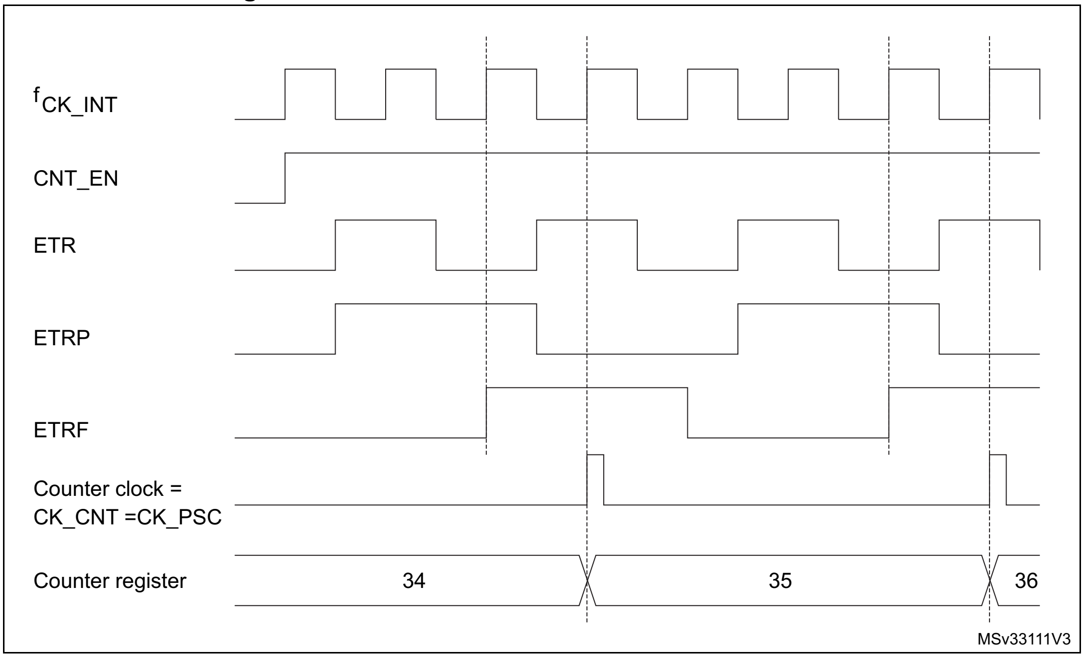
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. Select the proper ETR source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0]=0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0]=01 in the TIMx_SMCR register
4. Select rising edge detection on the ETR pin by writing ETP=0 in the TIMx_SMCR register
5. Enable external clock mode 2 by writing ECE=1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user should apply a division of the external signal by a proper ETPS prescaler setting.

Figure 396. Control circuit in external clock mode 2



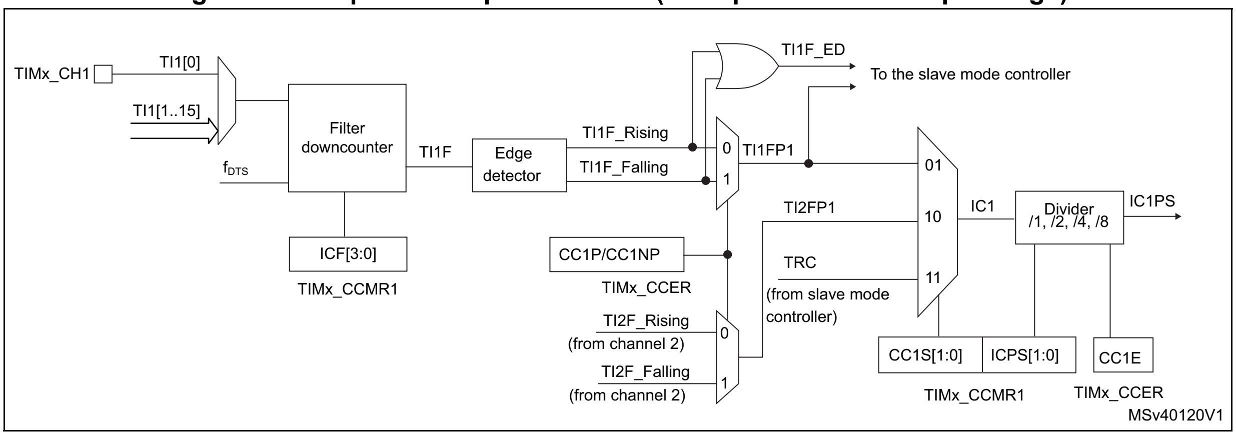
39.3.4 Capture/Compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 397. Capture/Compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 398. Capture/Compare channel 1 main circuit

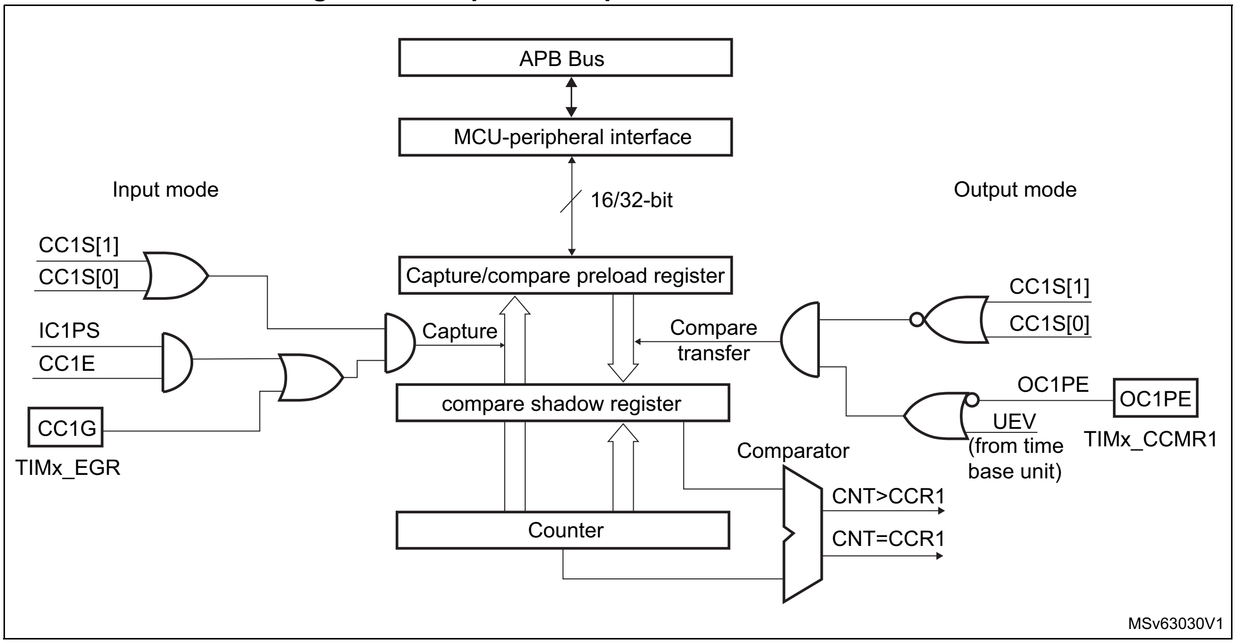
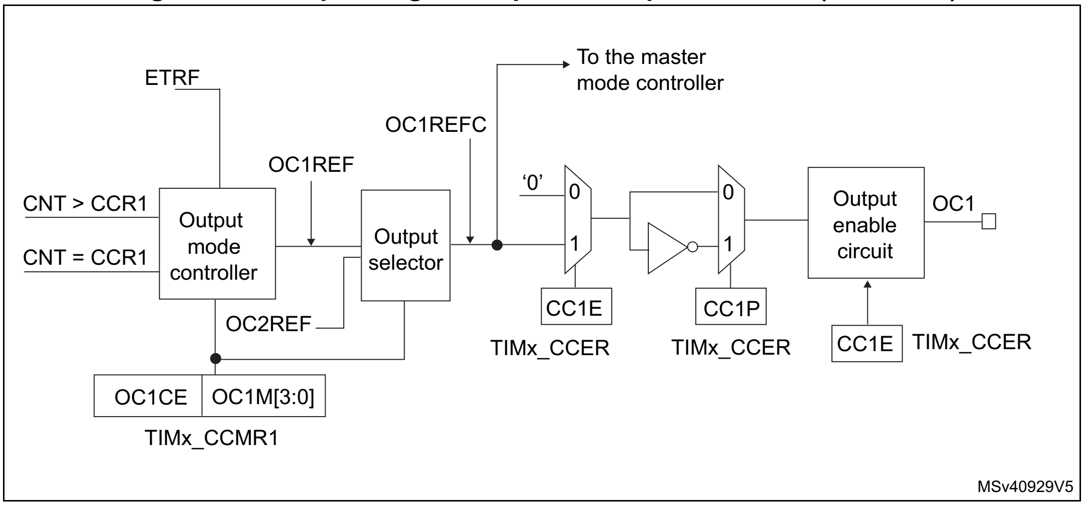


Figure 399. Output stage of Capture/Compare channel (channel 1)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

39.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing the CC1P and CC1NP and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

39.3.6 PWM input mode

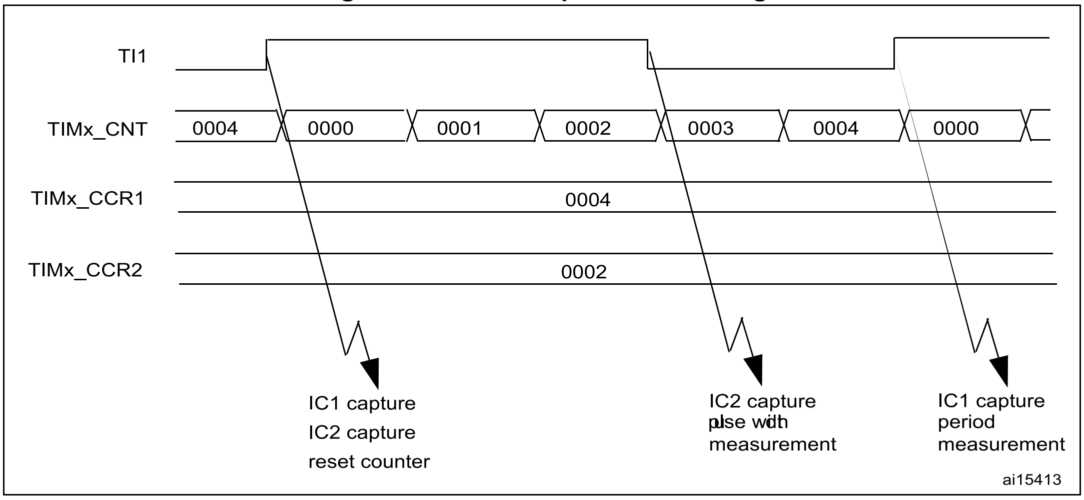
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to ‘0’ and the CC1NP bit to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P bit to ‘1’ and the CC2NP bit to ’0’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1 in the TIMx_CCER register.

Figure 400. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

39.3.7 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (ocxref/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus ocxref is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

e.g.: CCxP=0 (OCx active high) => OCx is forced to high level.

ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare Mode section.

39.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

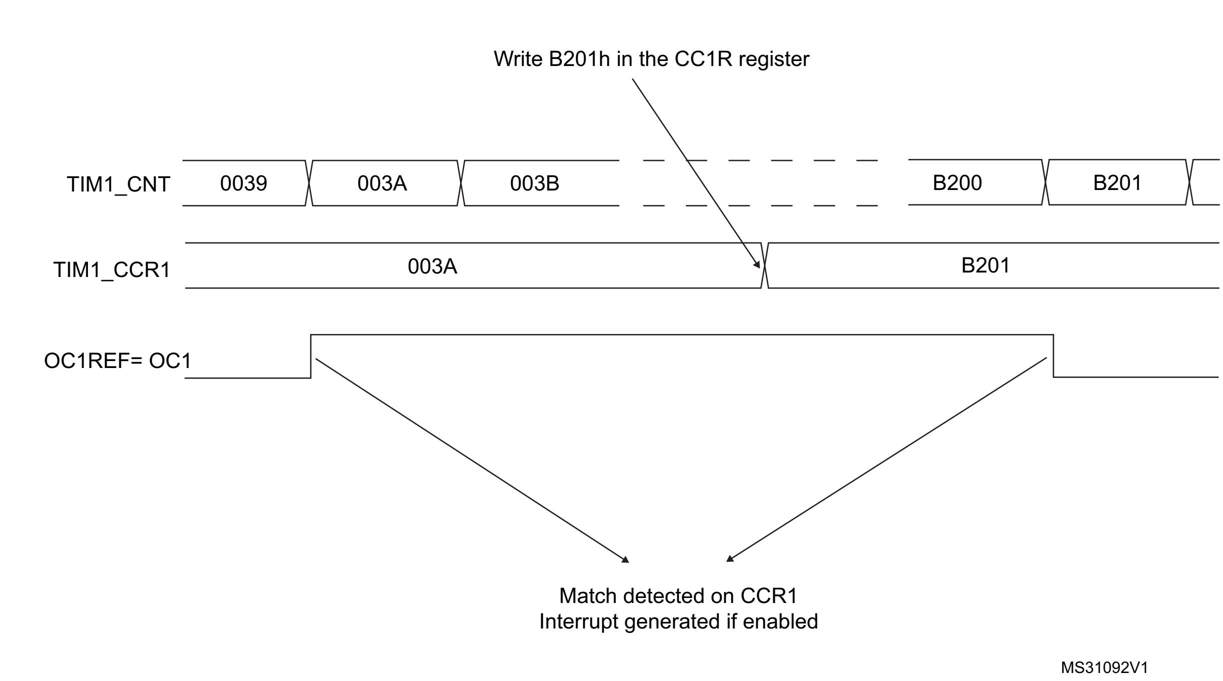
In output compare mode, the update event UEV has no effect on ocxref and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example, one must write OCxM=011, OCxPE=0, CCxP=0 and CCxE=1 to toggle OCx output pin when CNT matches CCRx, CCRx preload is not used, OCx is enabled and active high.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE=0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 401*.

Figure 401. Output compare mode, toggle on OC1



39.3.9 PWM mode

Pulse width modulation mode permits to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the OCREF_CLR functionality (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

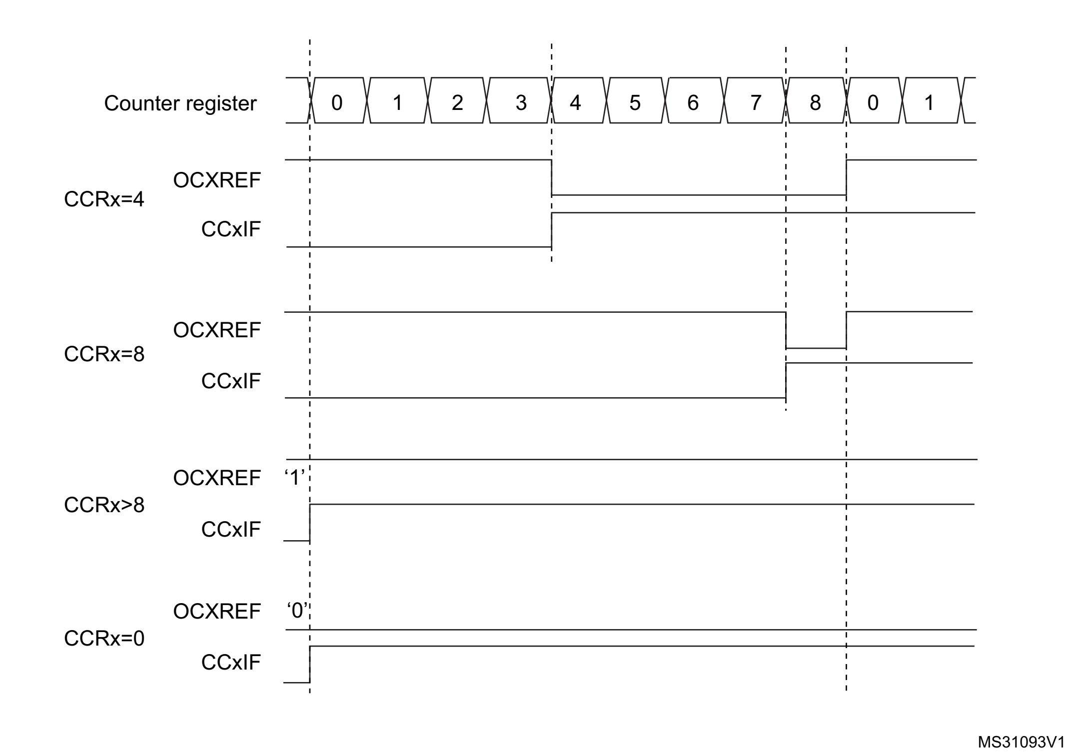
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode on page 2008*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at ‘1. If the compare value is 0 then OCxREF is held at ‘0. *Figure 402* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 402. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to *Downcounting mode on page 2011*.

In PWM mode 1, the reference signal ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, then ocxref is held at 100%. PWM is not possible in this mode.

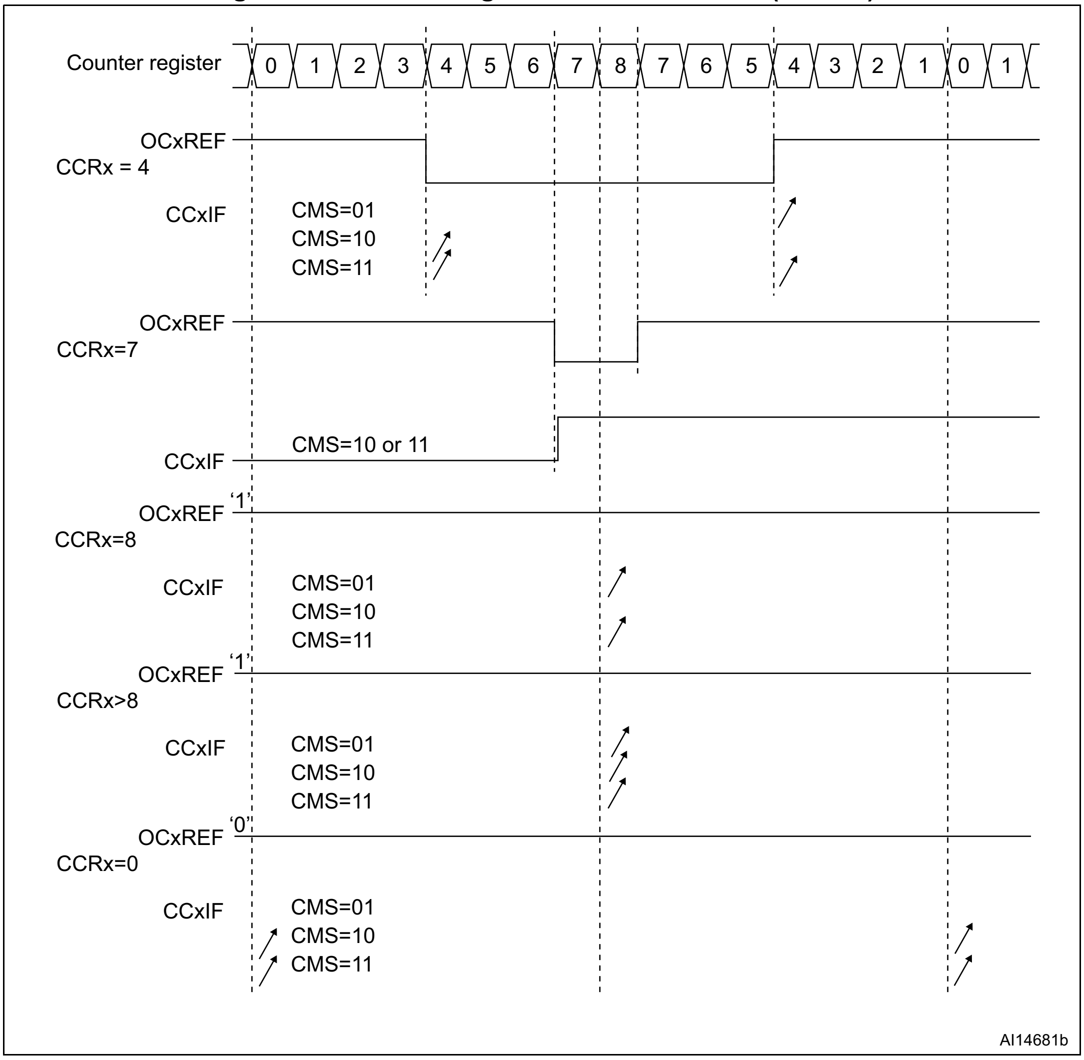
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from ‘00 (all the remaining configurations having the same effect on the ocxref/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 2014*.

Figure 403 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in TIMx_CR1 register.

Figure 403. Center-aligned PWM waveforms (ARR=8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the auto-reload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

39.3.10 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2
- OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

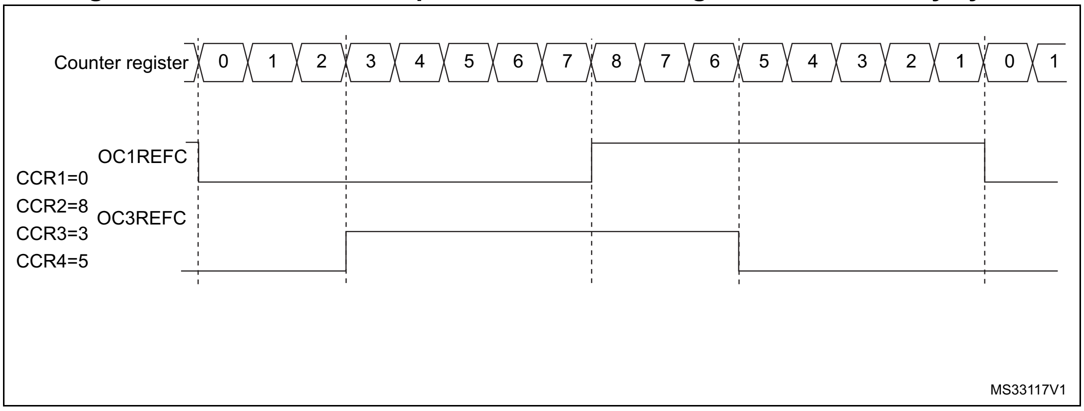
Asymmetric PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1110’ (Asymmetric PWM mode 1) or ‘1111’ (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an OC1REFC signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the OC2REF signal on channel 2, or an OC2REFC signal resulting from asymmetric PWM mode 2.

Figure 404 shows an example of signals that can be generated using Asymmetric PWM mode (channels 1 to 4 are configured in Asymmetric PWM mode 1).

Figure 404. Generation of 2 phase-shifted PWM signals with 50% duty cycle



39.3.11 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

– OC1REFC (or OC2REFC) is controlled by TIMx_CCR1 and TIMx_CCR2

– OC3REFC (or OC4REFC) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing ‘1100’ (Combined PWM mode 1) or ‘1101’ (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

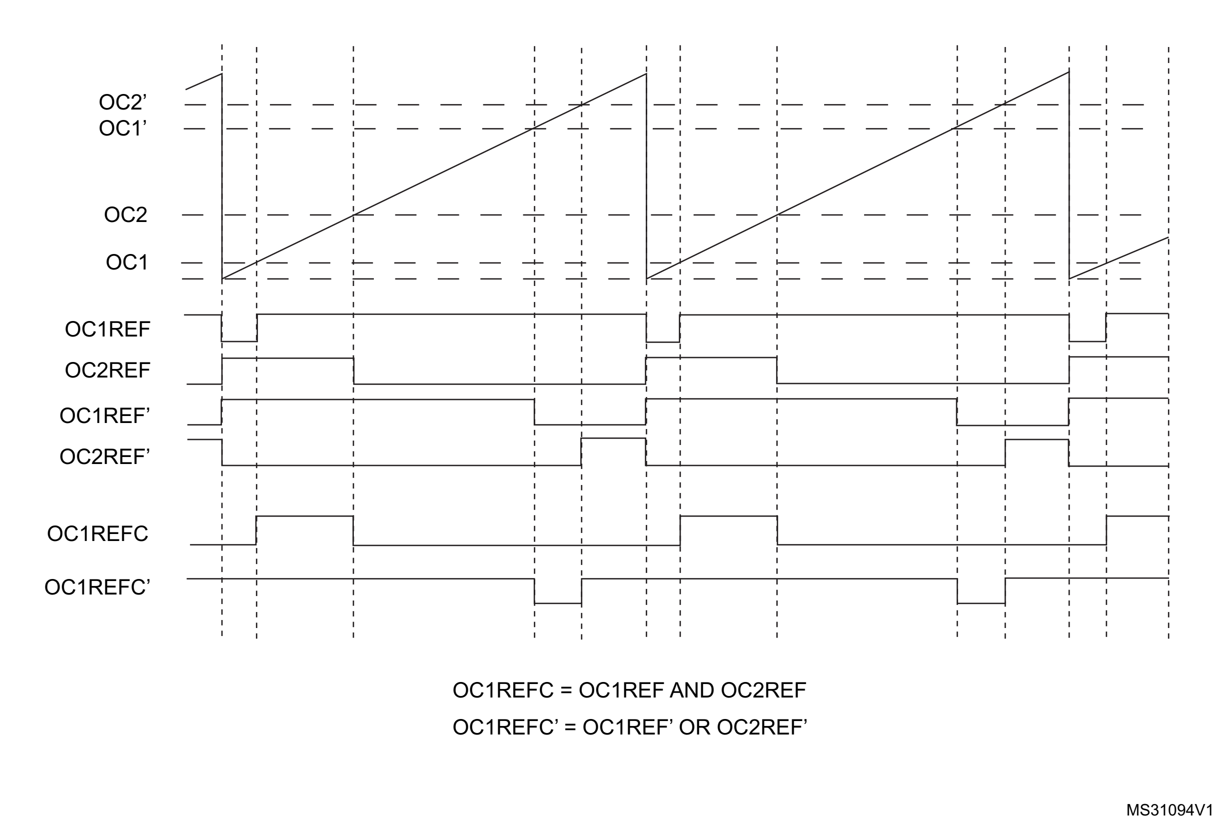
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 405 shows an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,
- Channel 3 is configured in Combined PWM mode 2,
- Channel 4 is configured in PWM mode 1

Figure 405. Combined PWM mode on channels 1 and 3



39.3.12 Clearing the OCxREF signal on an external event

The OCxREF signal of a given channel can be cleared when a high level is applied on the ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). OCxREF remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The ocref_clr_int is connected to the ETRF signal (ETR after filtering).

The OCxREF signal for a given channel can be reset by applying a high level on the ETRF input (OCxCE enable bit set to 1 in the corresponding TIMx_CCMRx register). OCxREF remains low until the next transition to the active state, on the following PWM cycle.

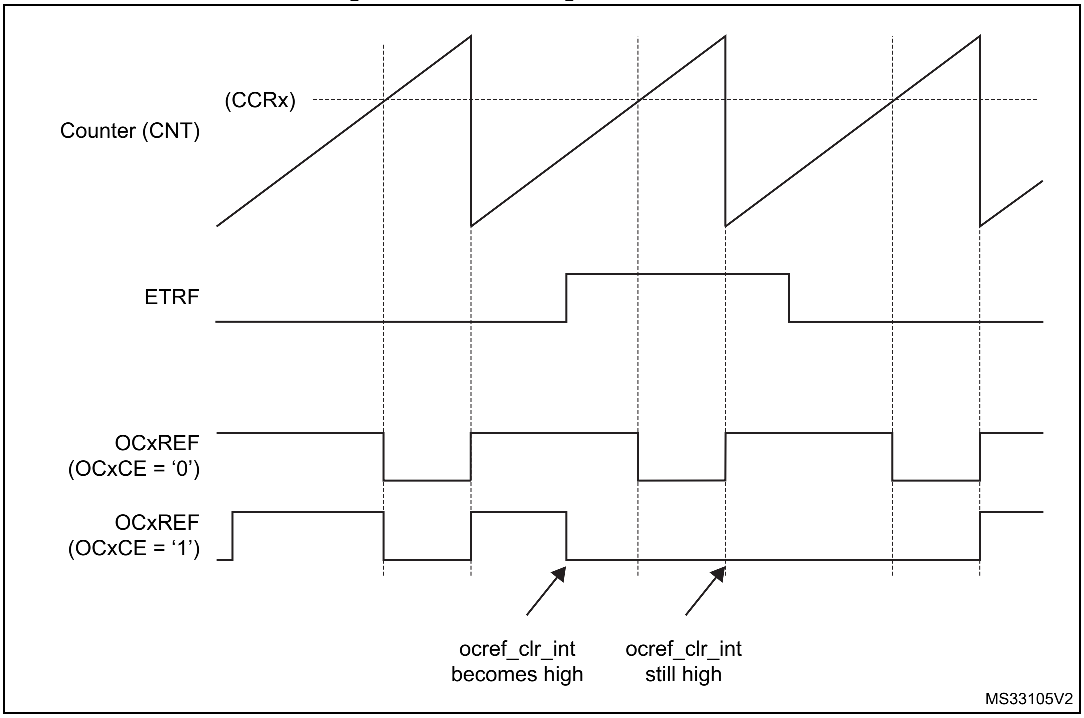
This function can be used only in the output compare and PWM modes. It does not work in forced mode.

For example, the OCxREF signal can be connected to the output of a comparator to be used for current handling. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 406 shows the behavior of the OCxREF signal when the ETRF input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 406. Clearing TIMx OCxREF



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), OCxREF is enabled again at the next counter overflow.

39.3.13 One-pulse mode

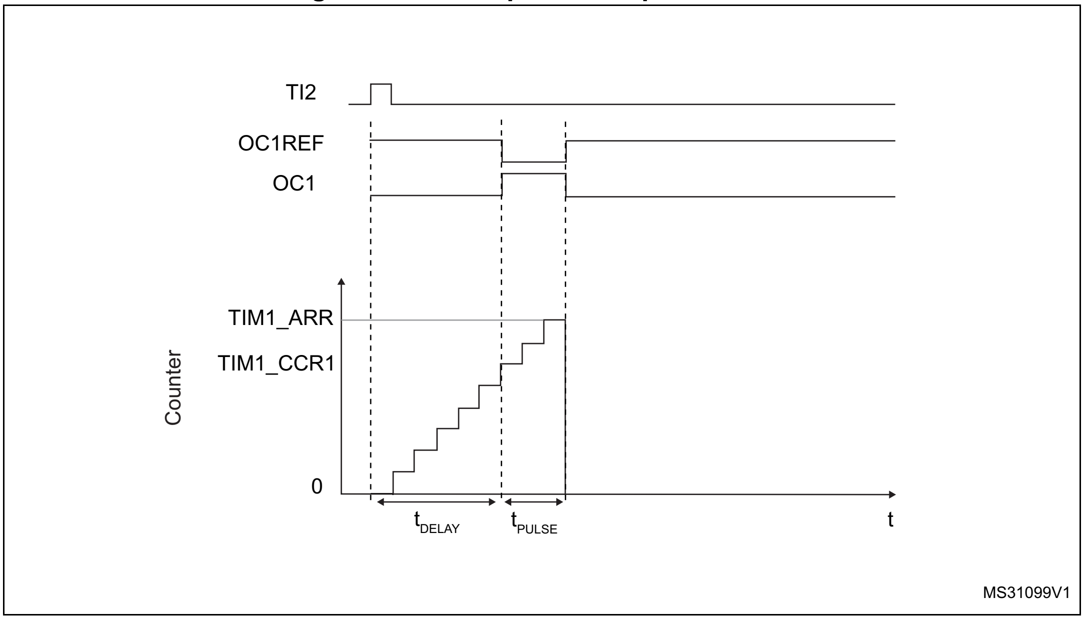
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT<CCRx ≤ ARR (in particular, 0<CCRx),

Figure 407. Example of one-pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let’s use TI2FP2 as trigger 1:

1. Select the proper TI2x source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 on TI2 by writing CC2S=01 in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P=0 and CC2NP=’0’ in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS=00110 in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to ‘110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0 to '1 when a compare match occurs and a transition from '1 to '0 when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0 in this example.

In our example, the DIR and CMS bits in the TIMx_CR1 register should be low.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

39.3.14 Retriggerable one pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 39.3.13*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = '1000' (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to '1000' or '1001' for Retriggerable OPM mode 1 or 2.

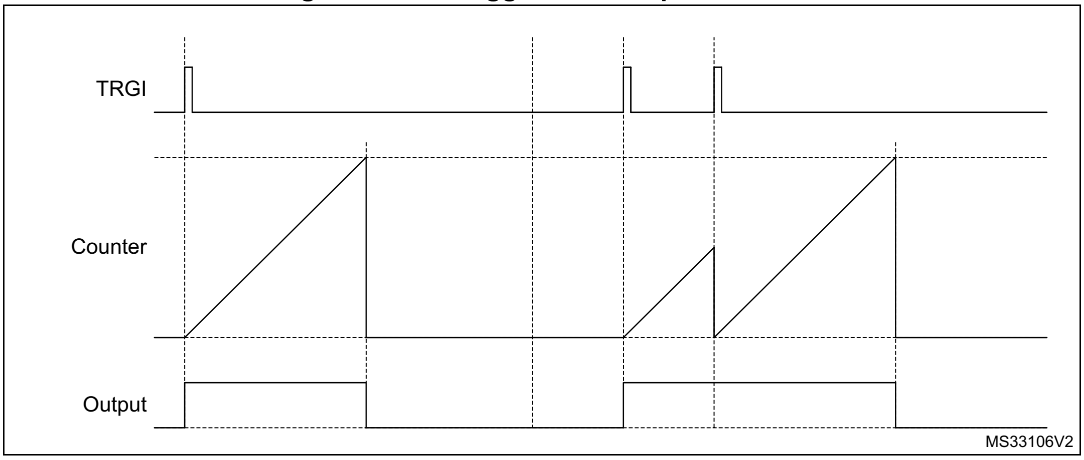
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode CCRx must be above or equal to ARR.

Note: In retriggerable one pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 408. Retriggerable one-pulse mode



39.3.15 Encoder interface mode

To select Encoder Interface mode write SMS='001 in the TIMx_SMCR register if the counter is counting on TI2 edges only, SMS=010 if it is counting on TI1 edges only and SMS=011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. Refer to *Table 282*. The counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1=TI1 if not filtered and not inverted, TI2FP2=TI2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to '1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever the counter is counting on TI1 only, TI2 only or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal.

In this mode, the counter is modified automatically following the speed and the direction of the-quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 282. Counting direction versus encoder signals

Active edge	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 409 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S= 01 (TIMx_CCMR1 register, TI1FP1 mapped on TI1)
- CC2S= 01 (TIMx_CCMR1 register, TI2FP2 mapped on TI2)
- CC1P and CC1NP = '0' (TIMx_CCER register, TI1FP1 noninverted, TI1FP1=TI1)
- CC2P and CC2NP = '0' (TIMx_CCER register, TI2FP2 noninverted, TI2FP2=TI2)
- SMS= 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges)
- CEN= 1 (TIMx_CR1 register, Counter is enabled)

Figure 409. Example of counter operation in encoder interface mode

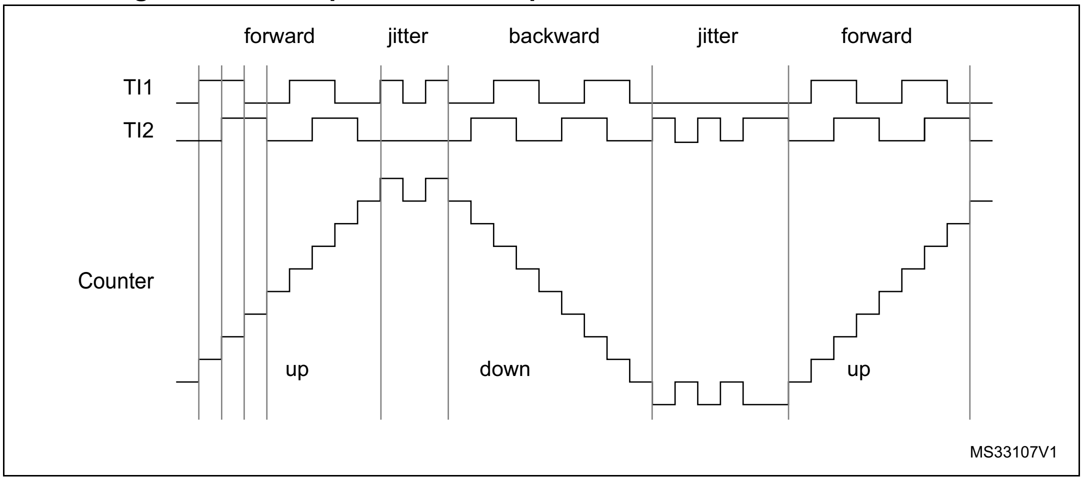
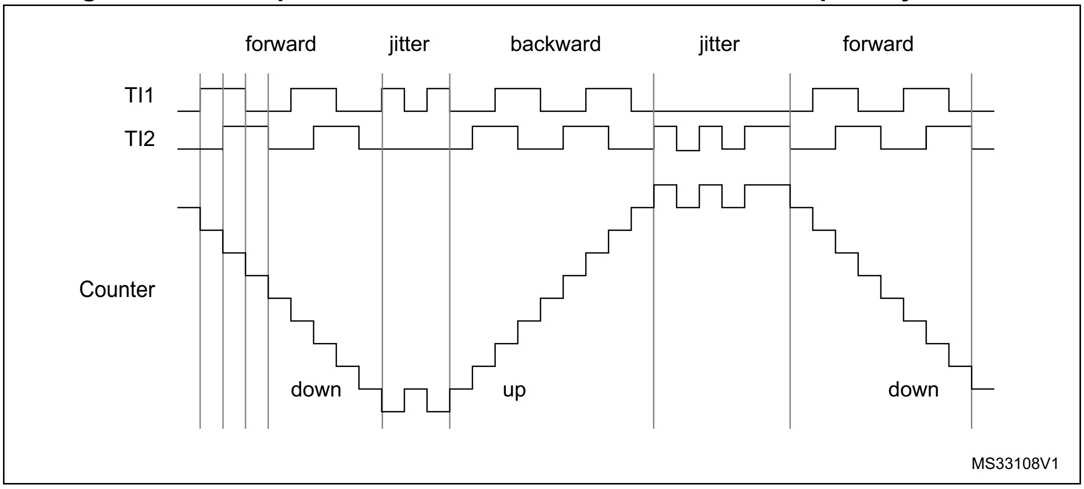


Figure 410 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P=1).

Figure 410. Example of encoder interface mode with TI1FP1 polarity inverted



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a Real-Time clock.

39.3.16 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This permits to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

39.3.17 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the three input pins TIMx_CH1 to TIMx_CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 38.3.25: Interfacing with Hall sensors on page 1952*.

39.3.18 Timers and external trigger synchronization

The TIMx Timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

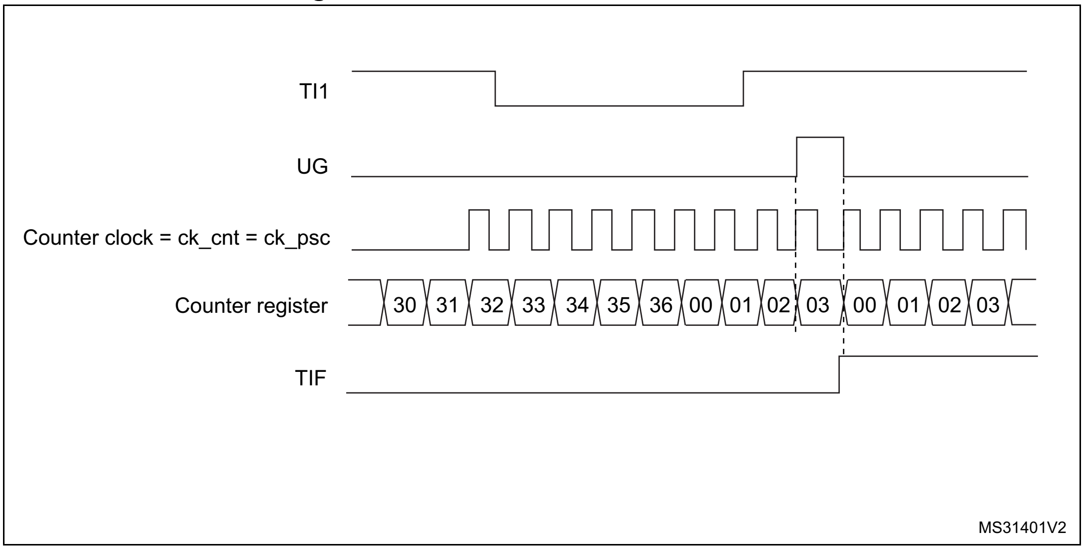
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 411. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

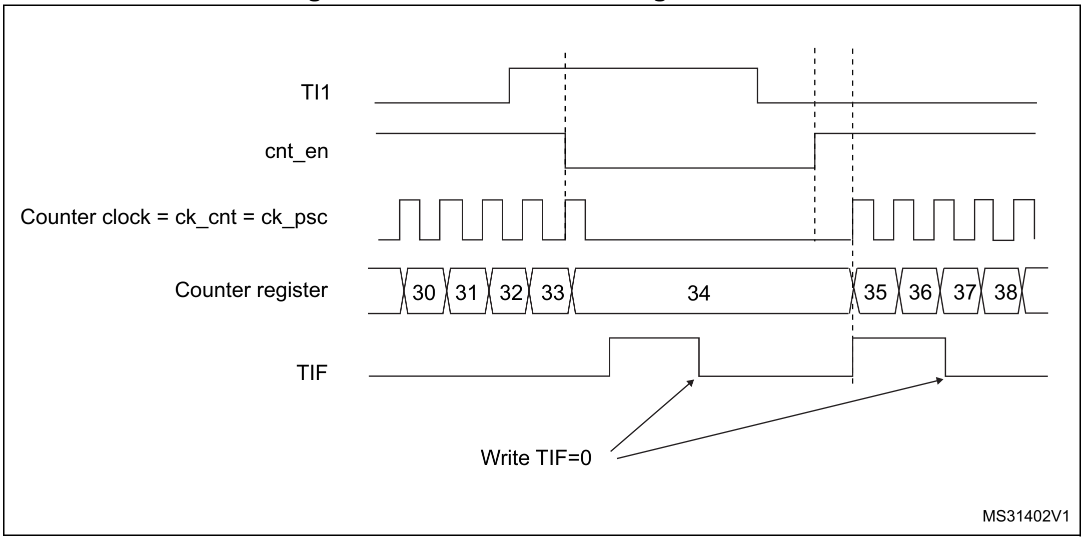
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 412. Control circuit in gated mode



1. The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Note: The configuration "CCxP=CCxNP=1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write

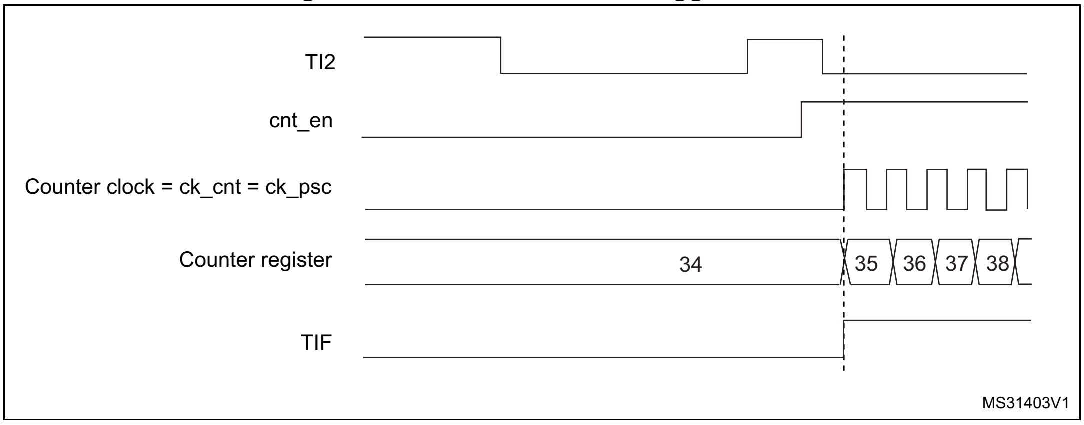
CC2P=1 and CC2NP=0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 413. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode: External Clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is recommended not to select ETR as TRGI through the TS bits of TIMx_SMCR register.

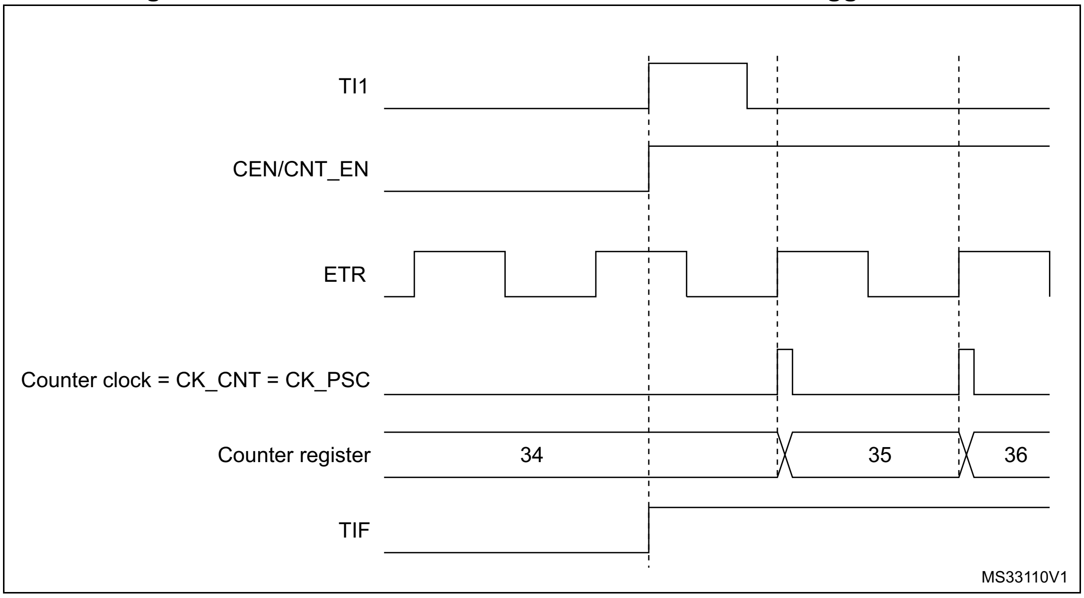
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   – ETF = 0000: no filter
   – ETPS=00: prescaler disabled
   – ETP=0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   – IC1F=0000: no filter.
   – The capture prescaler is not used for triggering and does not need to be configured.
   – CC1S=01in TIMx_CCMR1 register to select only the input capture source
   – CC1P=0 and CC1NP=0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the TIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 414. Control circuit in external clock mode 2 + trigger mode



39.3.19 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one Timer is configured in Master Mode, it can reset, start, stop or clock the counter of another Timer configured in Slave Mode.

Figure 415: Master/Slave timer example and *Figure 416: Master/slave connection example with 1 channel only timers* present an overview of the trigger selection and the master mode selection blocks.

Figure 415. Master/Slave timer example

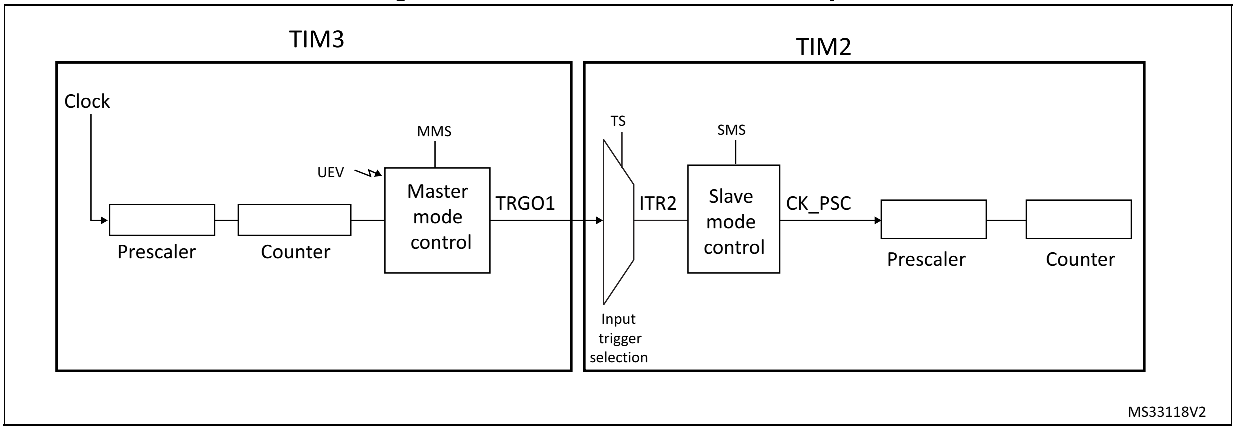
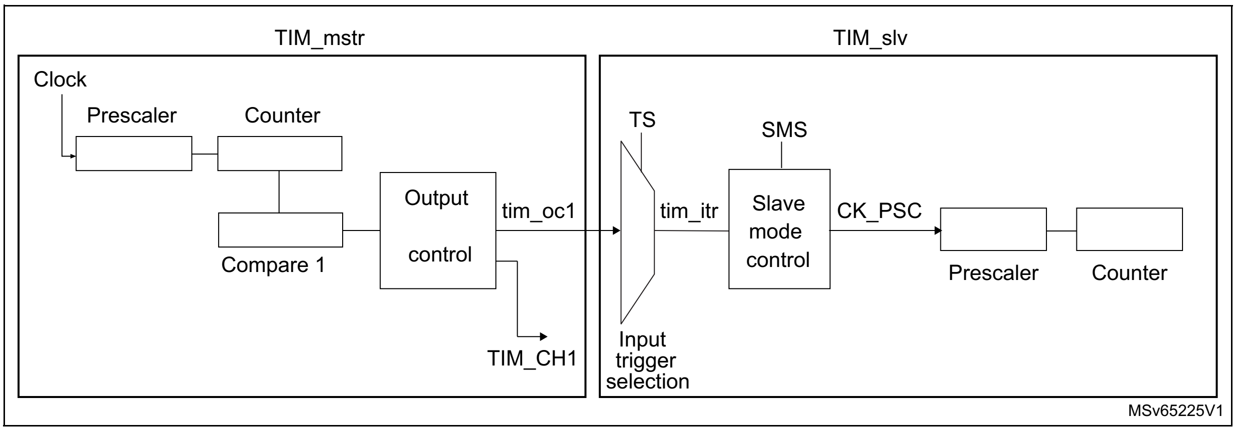


Figure 416. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 416) do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the "TIMx internal trigger connection" table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.
The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

Using one timer as prescaler for another timer

For example, TIM3 can be configured to act as a prescaler for TIM2. Refer to *Figure 415*. To do this:

1. Configure TIM3 in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS=010 is written in the TIM3_CR2 register, a rising edge is output on TRGO each time an update event is generated.
2. To connect the TRGO output of TIM3 to TIM2, TIM2 must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM2_SMCR register (writing TS=00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS=111 in the TIM2_SMCR register). This causes TIM2 to be clocked by the rising edge of the periodic TIM3 trigger signal (which correspond to the TIM3 counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If OCx is selected on TIM3 as the trigger output (MMS=1xx), its rising edge is used to clock the counter of TIM2.

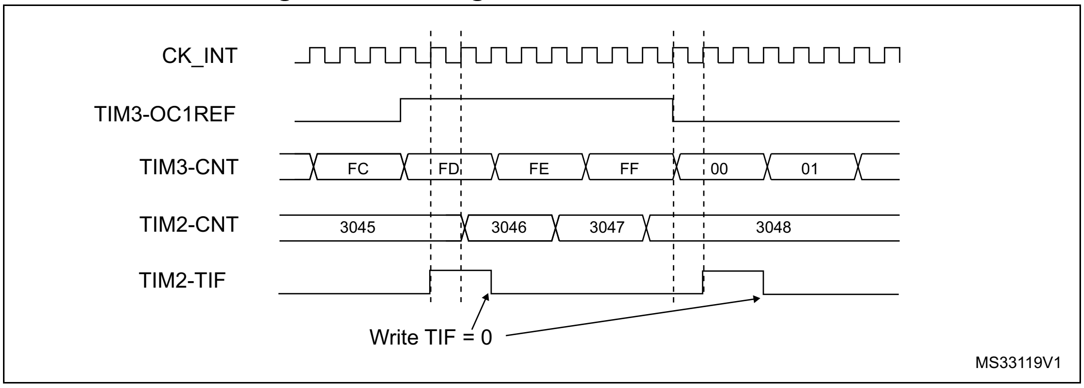
Using one timer to enable another timer

In this example, we control the enable of TIM2 with the output compare 1 of Timer 3. Refer to *Figure 415* for connections. TIM2 counts on the divided internal clock only when OC1REF of TIM3 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
6. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the TIM2 counter enable signal.

Figure 417. Gating TIM2 with OC1REF of TIM3

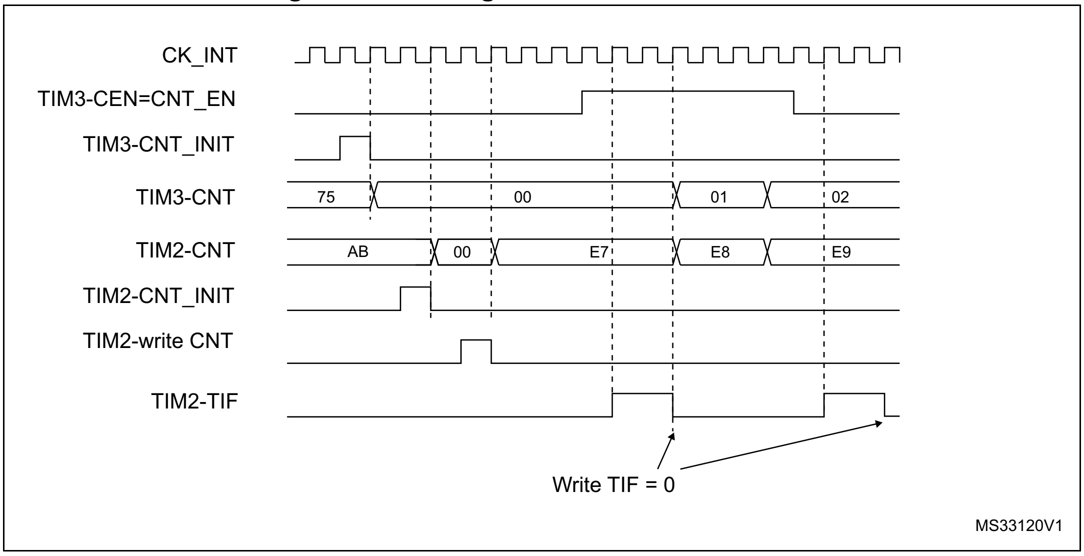


In the example in *Figure 417*, the TIM2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM3. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 418*), we synchronize TIM3 and TIM2. TIM3 is the master and starts from 0. TIM2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM2 stops when TIM3 is disabled by writing ‘0 to the CEN bit in the TIM3_CR1 register:

1. Configure TIM3 master mode to send its Output Compare 1 Reference (OC1REF) signal as trigger output (MMS=100 in the TIM3_CR2 register).
2. Configure the TIM3 OC1REF waveform (TIM3_CCMR1 register).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in gated mode (SMS=101 in TIM2_SMCR register).
5. Reset TIM3 by writing ‘1 in UG bit (TIM3_EGR register).
6. Reset TIM2 by writing ‘1 in UG bit (TIM2_EGR register).
7. Initialize TIM2 to 0xE7 by writing ‘0xE7’ in the TIM2 counter (TIM2_CNTL).
8. Enable TIM2 by writing ‘1 in the CEN bit (TIM2_CR1 register).
9. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).
10. Stop TIM3 by writing ‘0 in the CEN bit (TIM3_CR1 register).

Figure 418. Gating TIM2 with Enable of TIM3

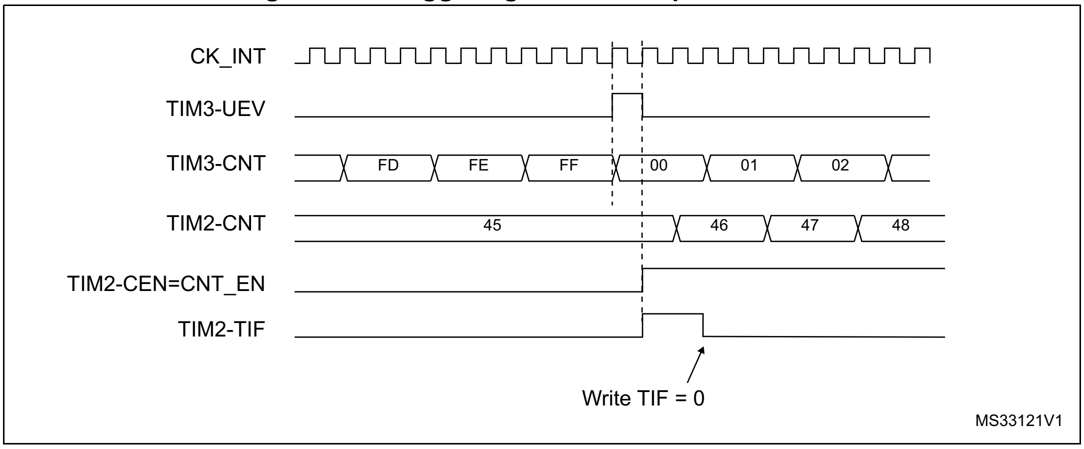


Using one timer to start another timer

In this example, we set the enable of Timer 2 with the update event of Timer 3. Refer to *Figure 415* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as the update event is generated by Timer 1. When Timer 2 receives the trigger signal its CEN bit is automatically set and the counter counts until we write ‘0 to the CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

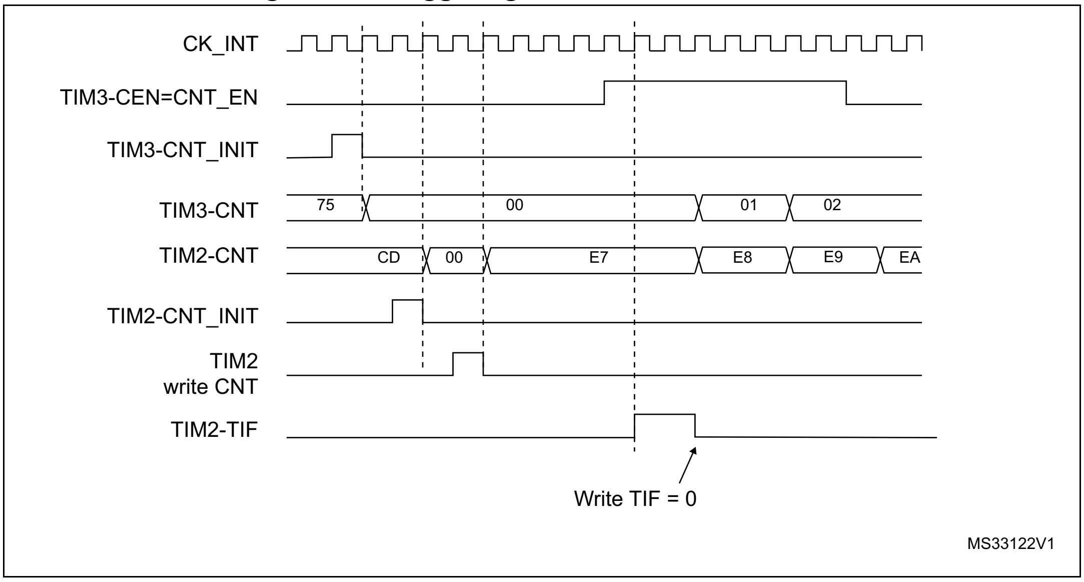
1. Configure TIM3 master mode to send its Update Event (UEV) as trigger output (MMS=010 in the TIM3_CR2 register).
2. Configure the TIM3 period (TIM3_ARR registers).
3. Configure TIM2 to get the input trigger from TIM3 (TS=00010 in the TIM2_SMCR register).
4. Configure TIM2 in trigger mode (SMS=110 in TIM2_SMCR register).
5. Start TIM3 by writing ‘1 in the CEN bit (TIM3_CR1 register).

Figure 419. Triggering TIM2 with update of TIM3



As in the previous example, both counters can be initialized before starting counting. *Figure 420* shows the behavior with the same configuration as in *Figure 419* but in trigger mode instead of gated mode (SMS=110 in the TIM2_SMCR register).

Figure 420. Triggering TIM2 with Enable of TIM3



Starting 2 timers synchronously in response to an external trigger

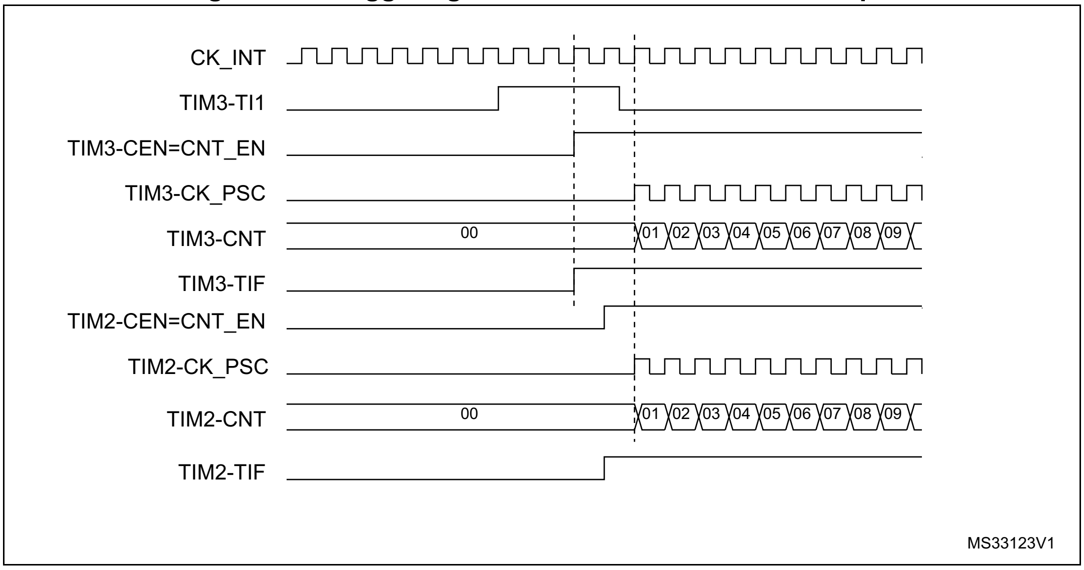
In this example, we set the enable of TIM3 when its TI1 input rises, and the enable of TIM2 with the enable of TIM3. Refer to *Figure 415* for connections. To ensure the counters are aligned, TIM3 must be configured in Master/Slave mode (slave with respect to TI1, master with respect to TIM2):

1. Configure TIM3 master mode to send its Enable as trigger output (MMS=001 in the TIM3_CR2 register).
2. Configure TIM3 slave mode to get the input trigger from TI1 (TS=00100 in the TIM3_SMCR register).
3. Configure TIM3 in trigger mode (SMS=110 in the TIM3_SMCR register).
4. Configure the TIM3 in Master/Slave mode by writing MSM=1 (TIM3_SMCR register).
5. Configure TIM2 to get the input trigger from TIM3 (TS=00000 in the TIM2_SMCR register).
6. Configure TIM2 in trigger mode (SMS=110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (TIM3), both counters starts counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode insert a delay between CNT_EN and CK_PSC on TIM3.

Figure 421. Triggering TIM3 and TIM2 with TIM3 TI1 input



Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

39.3.20 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to re-program part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

39.3.21 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on TIMx configuration bit in DBGMCU module. For more details, refer to *Section 64.10.9: Microprocessor debug unit (DBGMCU)*.

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_APB1FZ2), the outputs are disabled.

39.4 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

39.4.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (ETR, TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter
1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

39.4.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input
1: The TIMx_CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)
See also *Section 38.3.25: Interfacing with Hall sensors on page 1952*

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits permit to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO)

100: **Compare** - OC1REFC signal is used as trigger output (TRGO)
101: **Compare** - OC2REFC signal is used as trigger output (TRGO)
110: **Compare** - OC3REFC signal is used as trigger output (TRGO)
111: **Compare** - OC4REFC signal is used as trigger output (TRGO)

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

39.4.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or $\overline{\text{ETR}}$ is used for trigger operations

0: ETR is non-inverted, active at high level or rising edge

1: ETR is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK_INT frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.
00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
00111: External Trigger input (ETRF)
01000: Internal Trigger 4 (ITR4)
01001: Internal Trigger 5 (ITR5)
01010: Internal Trigger 6 (ITR6)
01011: Internal Trigger 7 (ITR7)
01100: Internal Trigger 8 (ITR8)
Others: Reserved
See *Table 283: TIMx internal trigger connection on page 2055* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (see Input Control register and Control Register description.
0000: Slave mode disabled - if CEN = ‘1 then the prescaler is clocked directly by the internal clock.
0001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
0010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
0011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS=00100). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 283. TIMx internal trigger connection

Slave TIM	ITR0	ITR1	ITR2	ITR3	ITR4	ITR5	ITR6	ITR7	ITR8
TIM2	TIM1	TIM8	TIM3	TIM4	-	-	-	-	-
TIM3	TIM1	TIM2	TIM15	TIM4	-	-	-	-	-
TIM4	TIM1	TIM2	TIM3	TIM8	-	-	-	-	-
TIM5	TIM1	TIM8	TIM3	TIM4	-	-	fdcan1_soc	-	OTG_SOF

39.4.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
- 0: Trigger DMA request disabled.
- 1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
- 0: CC4 DMA request disabled.
- 1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
- 0: CC3 DMA request disabled.
- 1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
- 0: CC2 DMA request disabled.
- 1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
- 0: CC1 DMA request disabled.
- 1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
- 0: Trigger interrupt disabled.
- 1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

39.4.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow (for TIM2 to TIM4) and if UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS=0 and UDIS=0 in the TIMx_CR1 register.

39.4.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (TIMx_ARR) if DIR=1 (downcounting).

39.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E=0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

39.4.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: OC1Ref is not affected by the ETRF input
1: OC1Ref is cleared as soon as a High level is detected on ETRF input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.
0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.
0100: Force inactive level - OC1REF is forced low.
0101: Force active level - OC1REF is forced high.
0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (OC1REF=‘0) as long as TIMx_CNT>TIMx_CCR1 else active (OC1REF=1).
0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.
1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.
1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.
1010: Reserved,
1011: Reserved,
1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.
1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.
1110: Asymmetric PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.
1111: Asymmetric PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC outputs OC1REF when the counter is counting up, OC2REF when it is counting down.

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.
01: CC1 channel is configured as input, IC1 is mapped on TI1.
10: CC1 channel is configured as input, IC1 is mapped on TI2.
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

39.4.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

39.4.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2) (x = 2 to 5)

Address offset: 0x1C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode

Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, IC4 is mapped on TI4
10: CC4 channel is configured as input, IC4 is mapped on TI3
11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode
Refer to OC1M description (bits 6:4 in TIMx_CCMR1 register)

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, IC3 is mapped on TI3
10: CC3 channel is configured as input, IC3 is mapped on TI4
11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

39.4.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 284. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard OCx channels depends on the OCx channel state and the GPIO control and alternate function registers.

39.4.12 TIMx counter (TIMx_CNT)(x = 2 to 5)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- This section is for UIFREMAP = 0
- Next section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CNT[31:16]**: Most significant part counter value (TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

39.4.13 TIMx counter [alternate] (TIMx_CNT)(x = 2 to 5)

Bit 31 of this register has two possible definitions depending on the value of UIFREMAP in TIMx_CR1 register:

- Previous section is for UIFREMAP = 0
- This section is for UIFREMAP = 1

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 **CNT[30:16]**: Most significant part counter value (TIM2 and TIM5)

Bits 15:0 **CNT[15:0]**: Least significant part of counter value

39.4.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

39.4.15 TIMx auto-reload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **ARR[31:16]**: High auto-reload value (TIM2 and TIM5)

Bits 15:0 **ARR[15:0]**: Low Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 39.3.1: Time-base unit on page 2006* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

39.4.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR1[31:16]**: High Capture/Compare 1 value (TIM2 and TIM5)

Bits 15:0 **CCR1[15:0]**: Low Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1). The TIMx_CCR1 register is read-only and cannot be programmed.

39.4.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR2[31:16]**: High Capture/Compare 2 value (TIM2 and TIM5)

Bits 15:0 **CCR2[15:0]**: Low Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

39.4.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR3[31:16]**: High Capture/Compare 3 value (TIM2 and TIM5)

Bits 15:0 **CCR3[15:0]**: Low Capture/Compare value

If channel CC3 is configured as output:
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC3 output.
If channel CC3is configured as input:
CCR3 is the counter value transferred by the last input capture 3 event (IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

39.4.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CCR4[31:16]**: High Capture/Compare 4 value (TIM2 and TIM5)

Bits 15:0 **CCR4[15:0]**: Low Capture/Compare value

1. if CC4 channel is configured as output (CC4S bits):
   CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
   The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC4 output.
2. if CC4 channel is configured as input (CC4S bits in TIMx_CCMR4 register):
   CCR4 is the counter value transferred by the last input capture 4 event (IC4). The TIMx_CCR4 register is read-only and cannot be programmed.

39.4.20 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the number of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...
Example: Let us consider the following transfer: DBL = 7 transfers & DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

39.4.21 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

39.4.22 TIM2 alternate function option register 1 (TIM2_AF1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection

These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: Reserved
0010: Reserved
0011: LSE
0100: SAI1 FS_A
0101: SAI1 FS_B
0110: eth_ptp_pps_out
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

39.4.23 TIM3 alternate function option register 1 (TIM3_AF1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reserved
0101: Reserved
0110: eth_ptp_pps_out
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

39.4.24 TIM4 alternate function option register 1 (TIM4_AF1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: ETR input is connected to I/O
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

39.4.25 TIM5 alternate function option register 1 (TIM5_AF1)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: ETR source selection
These bits select the ETR input source.
0000: ETR input is connected to I/O
0001: SAI2 FS_A connected to ETR input
0010: SAI2 FS_B connected to ETR input
0011: OTG_SOF
Others: Reserved

Bits 13:0 Reserved, must be kept at reset value.

39.4.26 TIM2 timer input selection register (TIM2_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM2_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM2_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM2_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM2_CH1 input
Others: Reserved

39.4.27 TIM3 timer input selection register (TIM3_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM3_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM3_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM3_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM3_CH1 input
Others: Reserved

39.4.28 TIM4 timer input selection register (TIM4_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM4_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM4_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection
These bits select the TI2[0] to TI2[15] input source.
0000: TIM4_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection
These bits select the TI1[0] to TI1[15] input source.
0000: TIM4_CH1 input
Others: Reserved

39.4.29 TIM5 timer input selection register (TIM5_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: TI4[0] to TI4[15] input selection
These bits select the TI4[0] to TI4[15] input source.
0000: TIM5_CH4 input
Others: Reserved

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: TI3[0] to TI3[15] input selection
These bits select the TI3[0] to TI3[15] input source.
0000: TIM5_CH3 input
Others: Reserved

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: TI2[0] to TI2[15] input selection

These bits select the TI2[0] to TI2[15] input source.
0000: TIM5_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: TI1[0] to TI1[15] input selection

These bits select the TI1[0] to TI1[15] input source.
0000: TIM5_CH1 input
0001: fdcan1_tmp
0010: fdcan1_rtp
Others: Reserved

39.4.30 TIMx register map

TIMx registers are mapped as described in the table below:

Table 285. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value																									0	0	0	0	0			
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0		0	0	0	0	0		0		0	0	0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0			0		0	0	0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**TIMx_CCMR2** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0

Table 285. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	TIMx_CNT	CNT[31] or UIFCPY	CNT[30:16] (TIM2 and TIM5 only, reserved on the other timers)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	ARR[31:16] (TIM2 and TIM5 only, reserved on the other timers)																ARR[15:0]															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved																																
0x34	TIMx_CCR1	CCR1[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	CCR2[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR2[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	CCR3[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR3[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	CCR4[31:16] (TIM2 and TIM5 only, reserved on the other timers)																CCR4[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved																																
0x48	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIM2_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x60	TIM3_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														

Table 285. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	**TIM4_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x60	**TIM5_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x68	**TIM2_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x68	**TIM3_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x68	**TIM4_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x68	**TIM5_TISEL**	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

40 General-purpose timers (TIM12/TIM13/TIM14)

40.1 TIM12/TIM13/TIM14 introduction

The TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 40.3.17: Timer synchronization (TIM12)*.

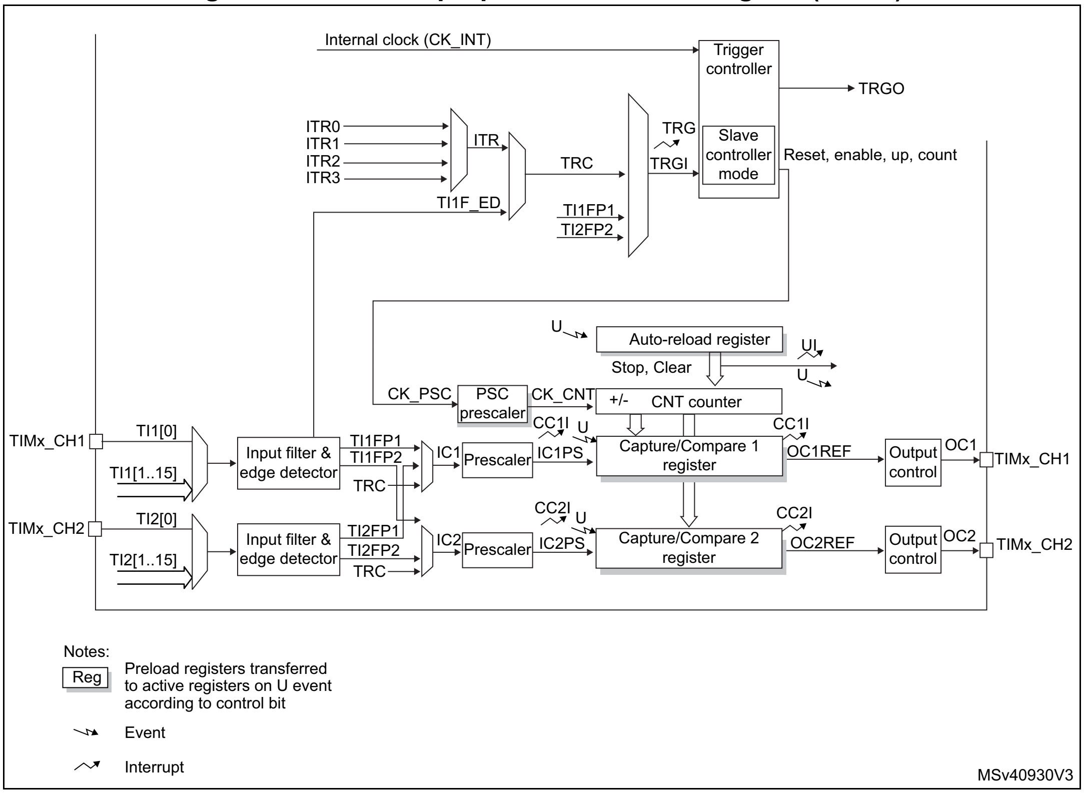
40.2 TIM12/TIM13/TIM14 main features

40.2.1 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization or count by internal trigger)
  - Input capture
  - Output compare

Figure 422. General-purpose timer block diagram (TIM12)

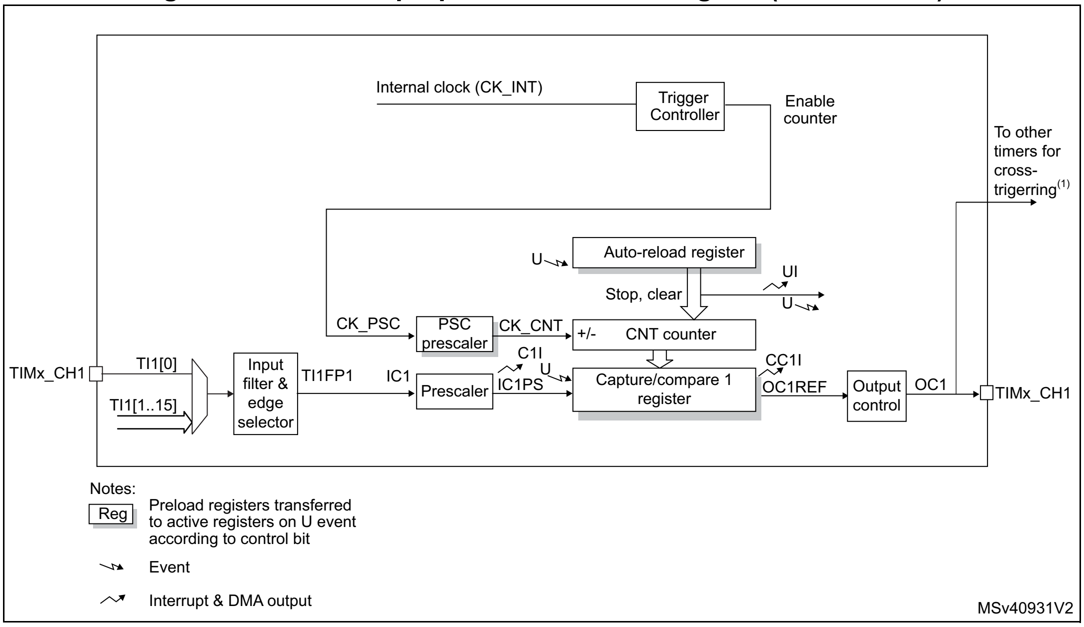


40.2.2 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

Figure 423. General-purpose timer block diagram (TIM13/TIM14)



1. This signal can be used as trigger for some slave timers, see *Section 40.3.18: Using timer output as trigger for other timers (TIM13/TIM14)*.

40.3 TIM12/TIM13/TIM14 functional description

40.3.1 Time-base unit

The main block of the timer is a 16-bit up-counter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 424 and *Figure 425* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 424. Counter timing diagram with prescaler division change from 1 to 2

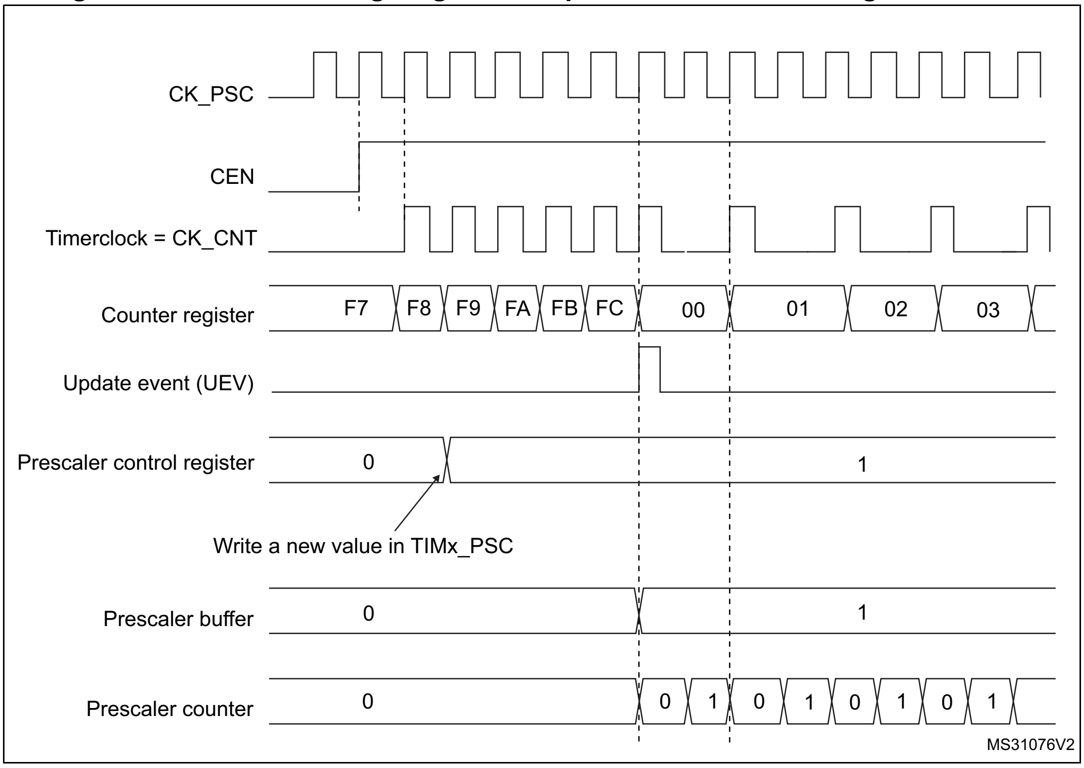
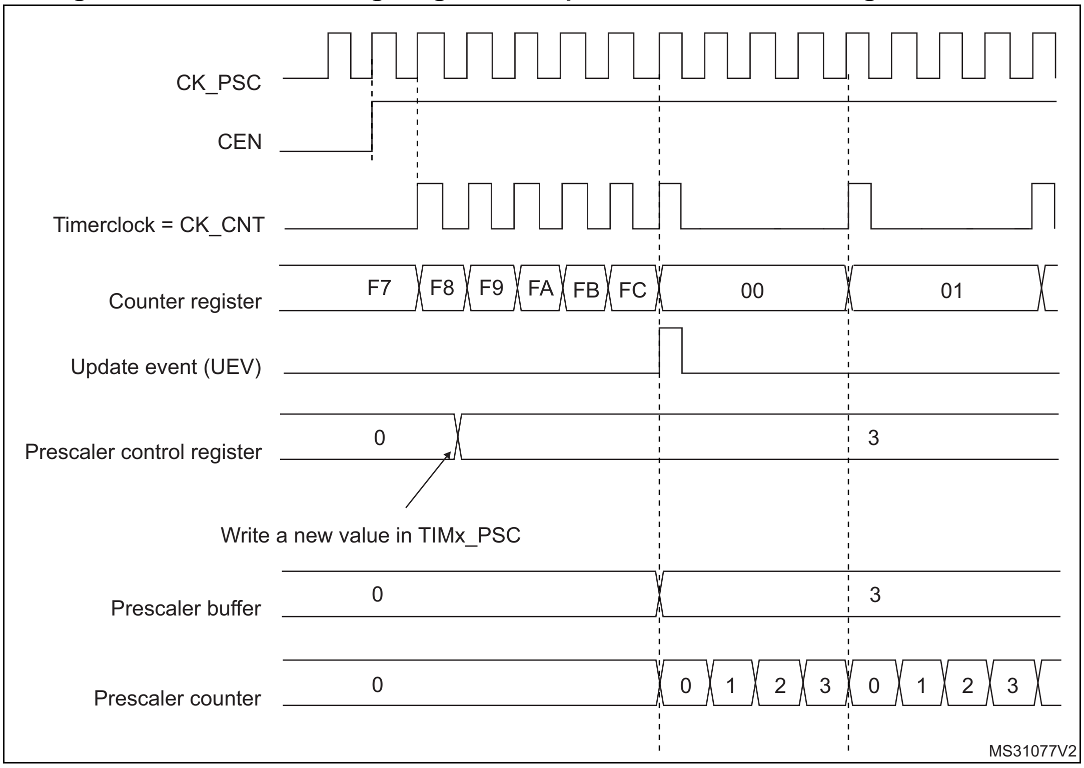


Figure 425. Counter timing diagram with prescaler division change from 1 to 4



40.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 426. Counter timing diagram, internal clock divided by 1

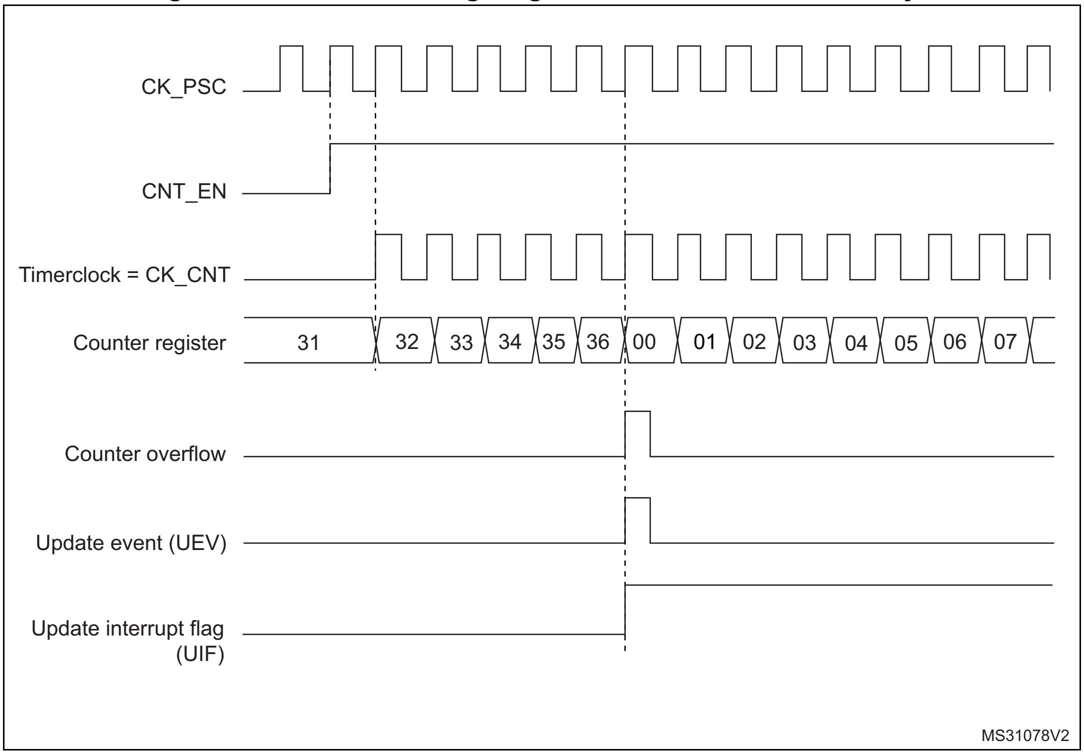


Figure 427. Counter timing diagram, internal clock divided by 2

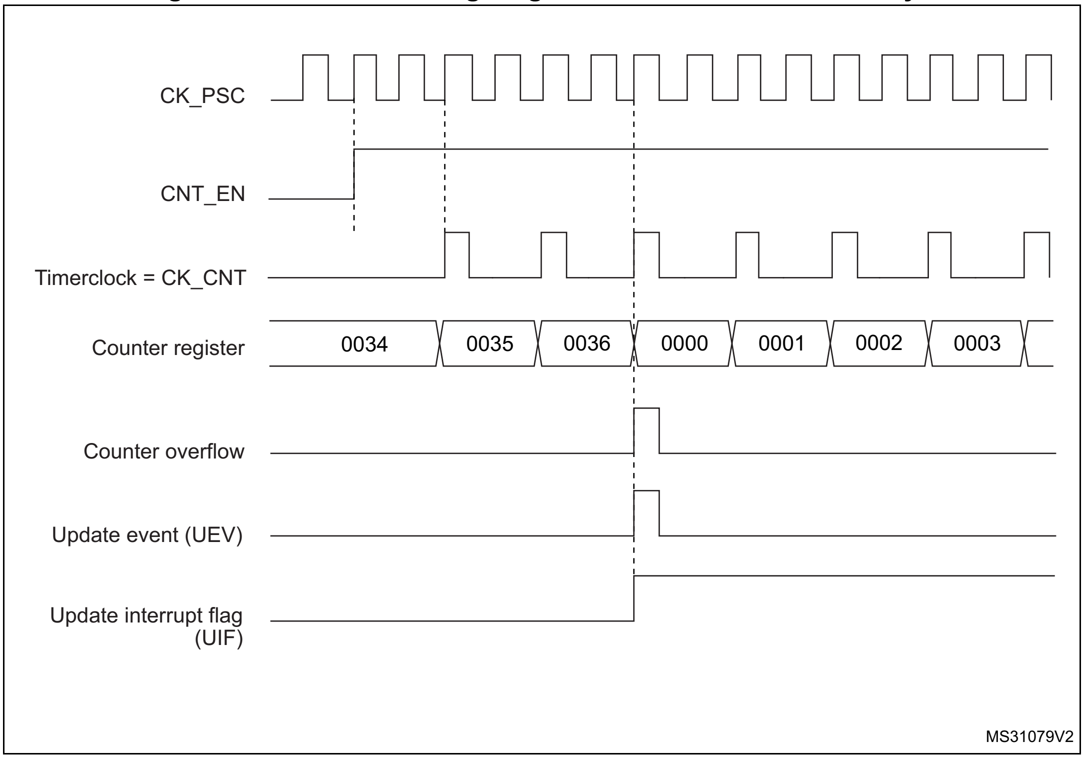


Figure 428. Counter timing diagram, internal clock divided by 4

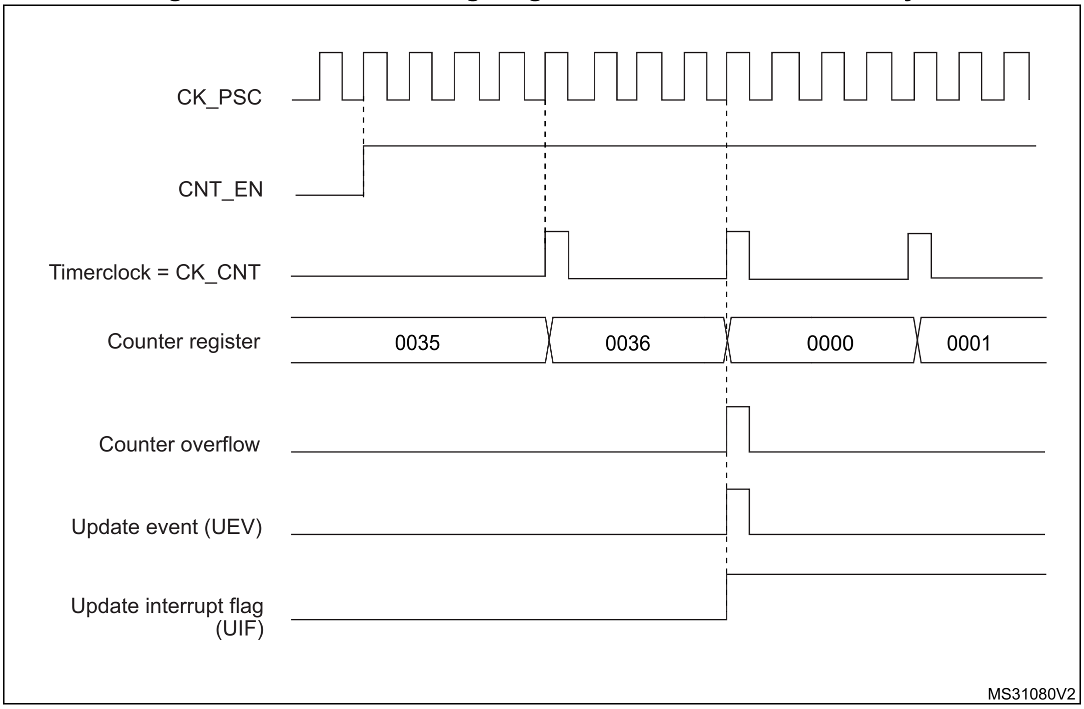


Figure 429. Counter timing diagram, internal clock divided by N

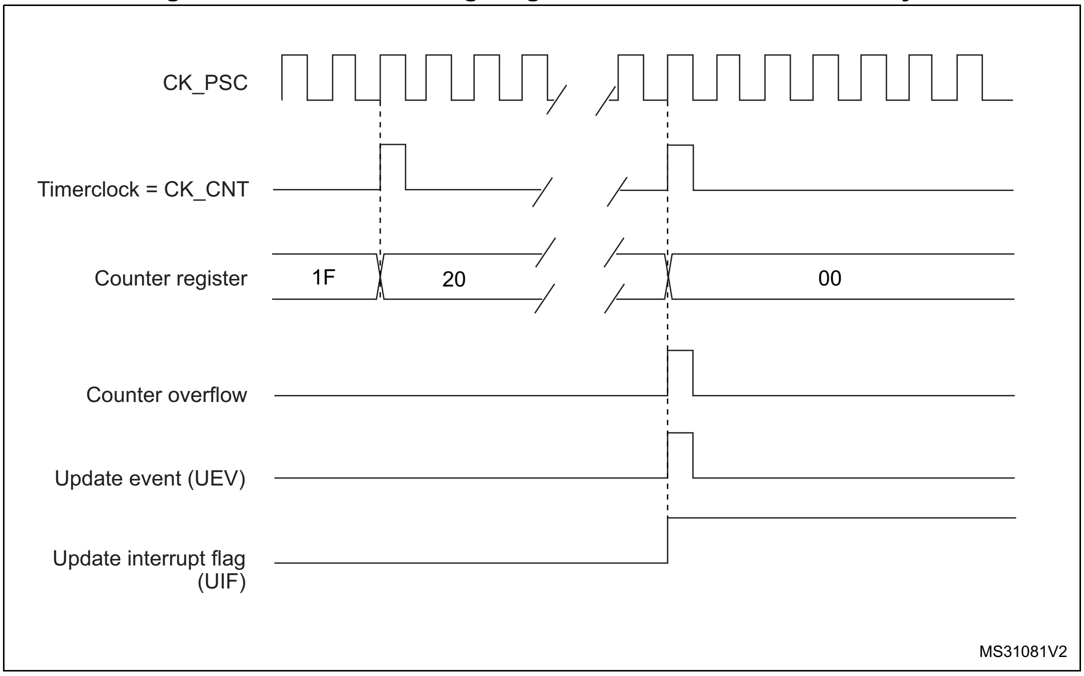


Figure 430. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

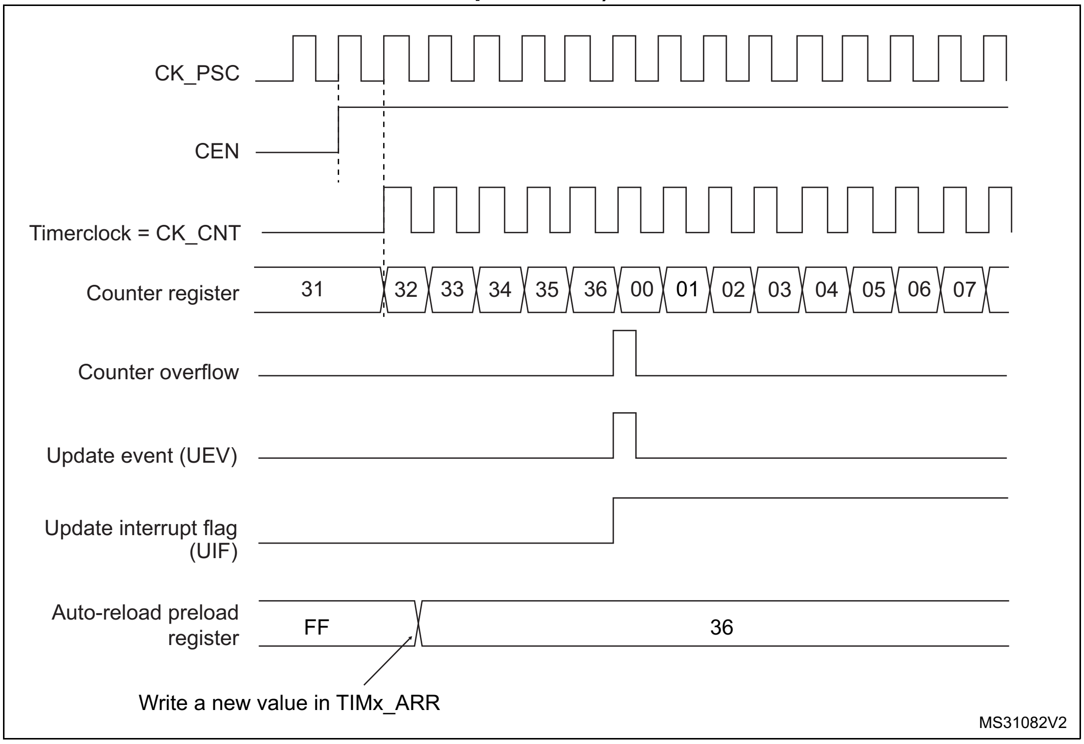
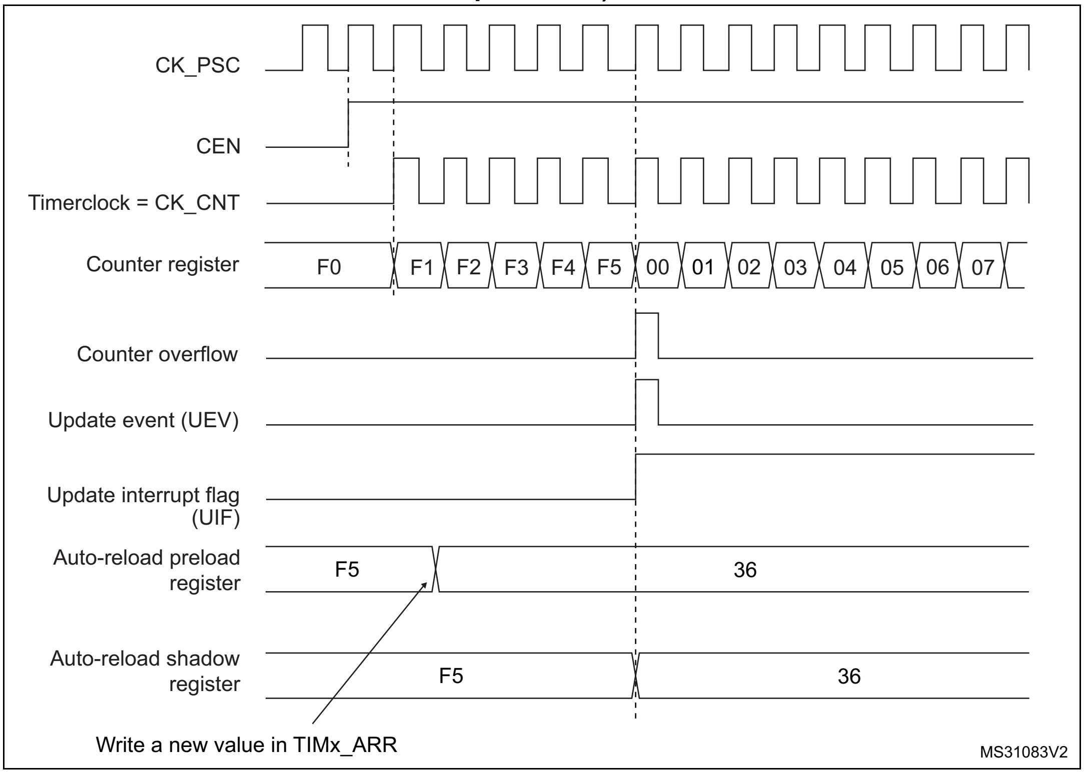


Figure 431. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



40.3.3 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (CK_INT)
- External clock mode1 (for TIM12): external input pin (TIx)
- Internal trigger inputs (ITRx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

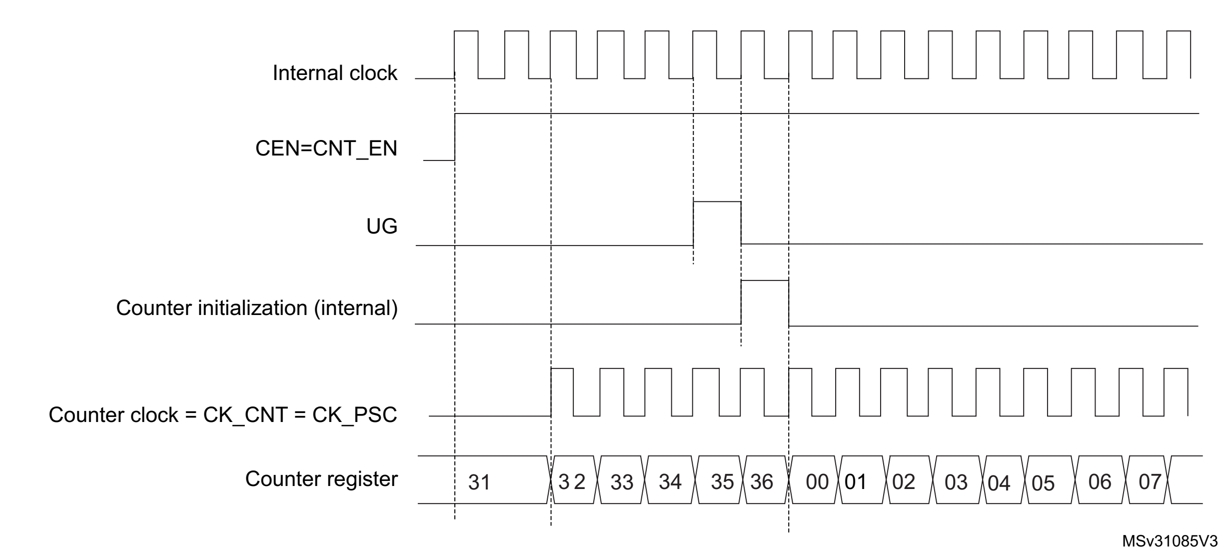
Internal clock source (CK_INT)

The internal clock source is the default clock source for TIM13/TIM14.

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS='000'). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 432 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

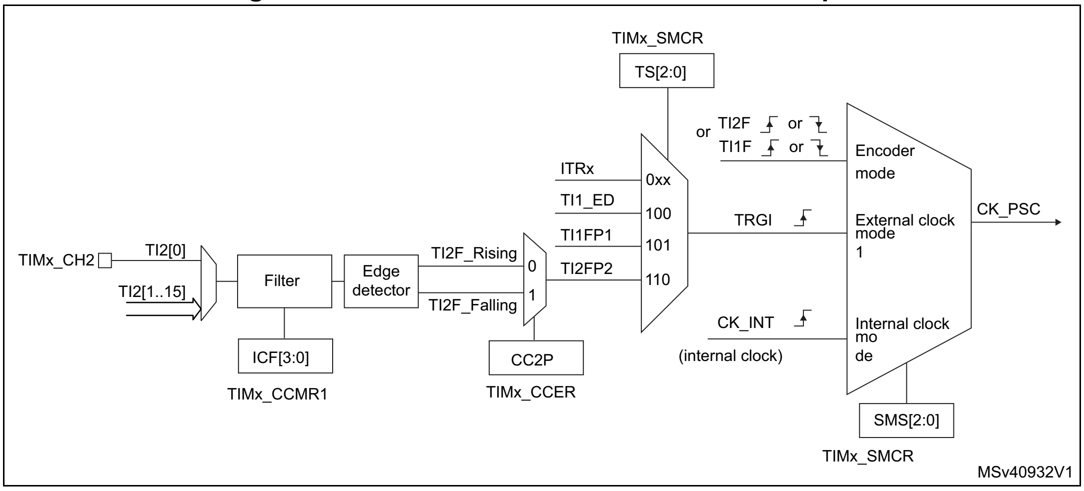
Figure 432. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1 (TIM12)

This mode is selected when SMS='111' in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 433. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

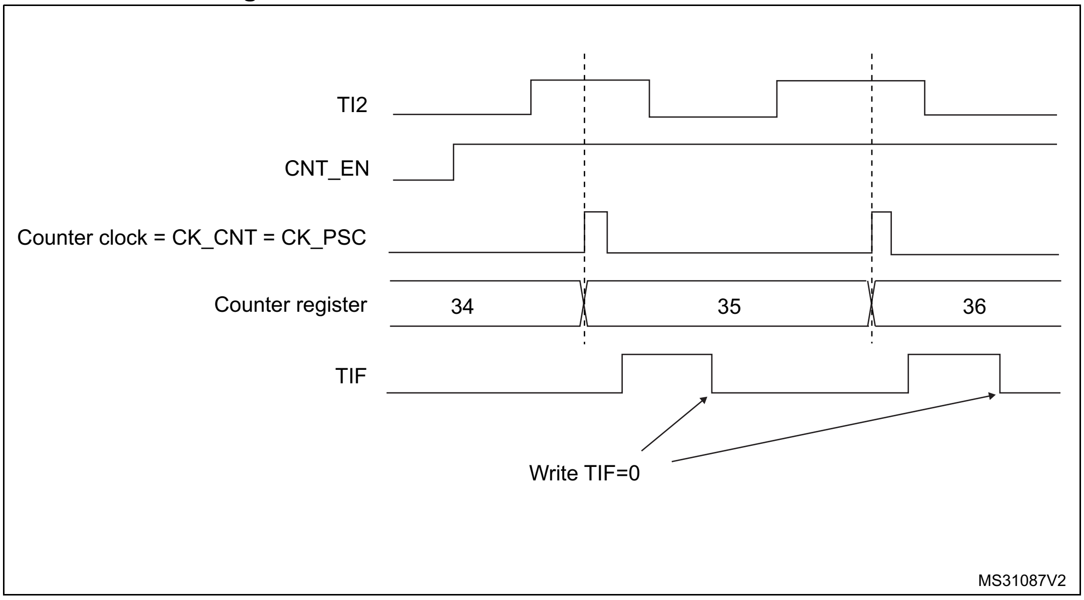
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=’0000’).
4. Select the rising edge polarity by writing CC2P=’0’ and CC2NP=’0’ in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=’111’ in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=’110’ in the TIMx_SMCR register.
7. Enable the counter by writing CEN=’1’ in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 434. Control circuit in external clock mode 1



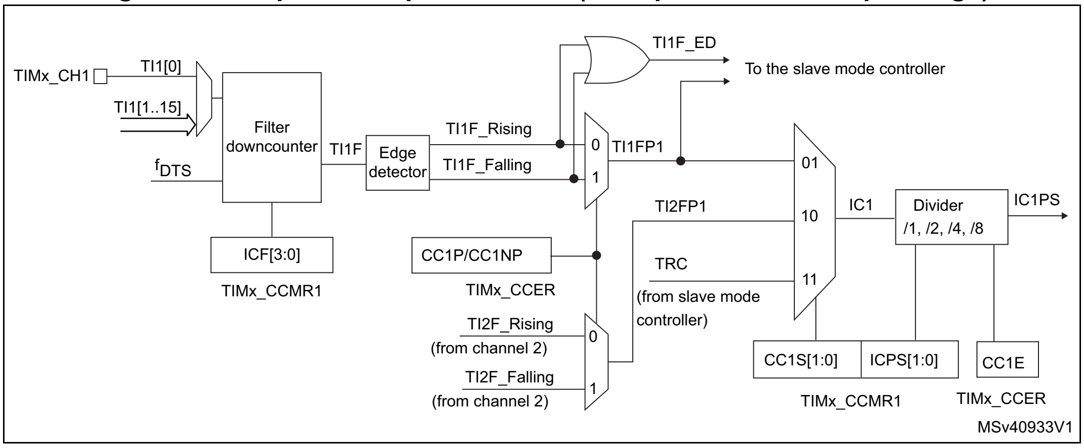
40.3.4 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 435 to *Figure 437* give an overview of one capture/compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 435. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 436. Capture/compare channel 1 main circuit

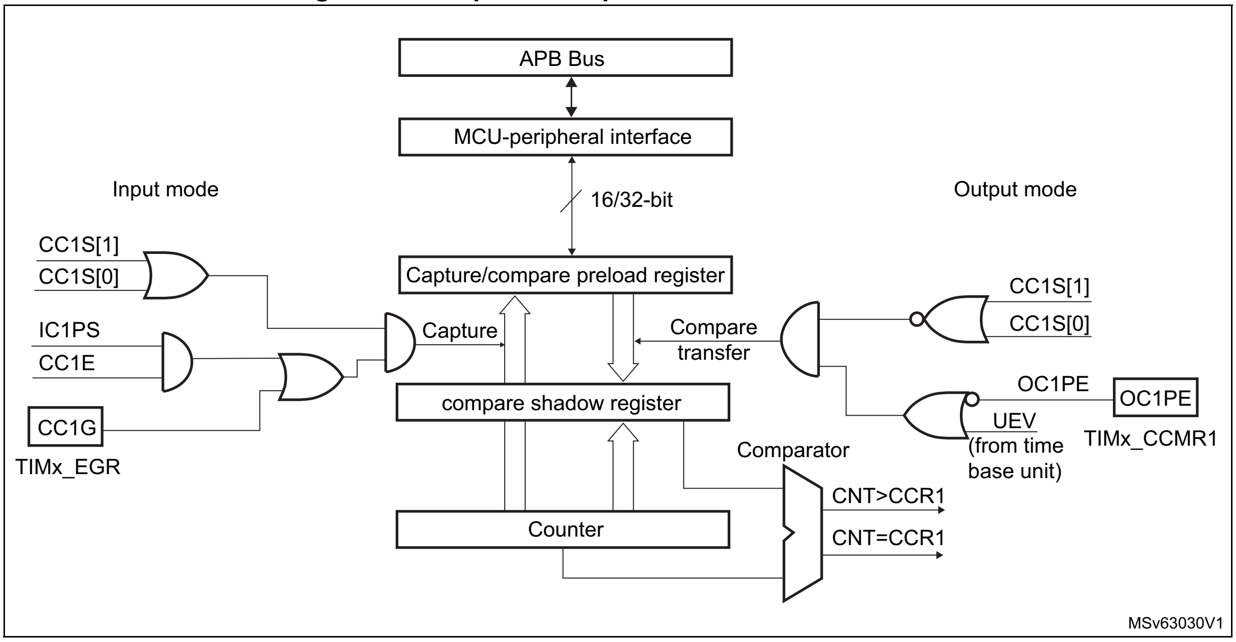
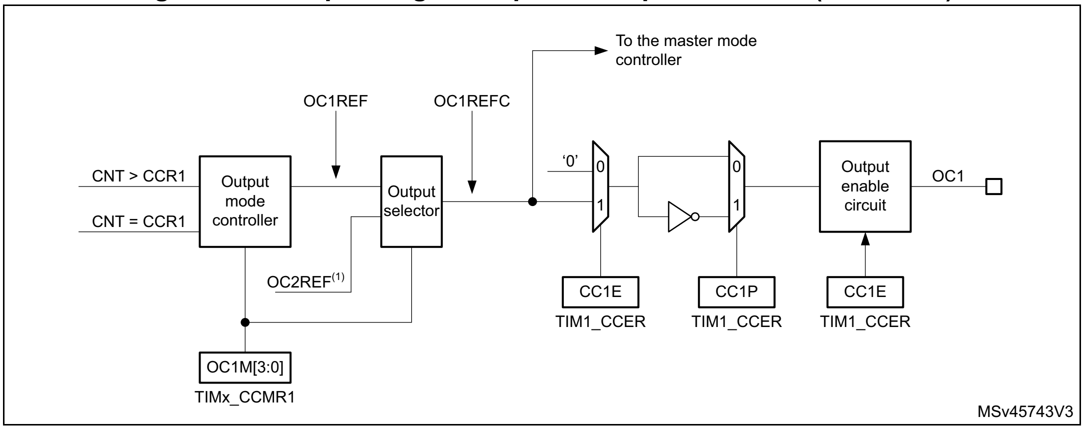


Figure 437. Output stage of capture/compare channel (channel 1)



1. Available on TIM12 only.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

40.3.5 Input capture mode

In Input capture mode, the Capture/Compare Registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to '01' in the TIMx_CCMR1 register. As soon as CC1S becomes different from '00', the channel is configured in input mode and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the TIx inputs). Let's imagine that, when toggling, the input signal is not stable during at must 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the

new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to ‘0011’ in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the TI1 channel by programming CC1P and CC1NP bits to ‘00’ in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

40.3.6 PWM input mode (only for TIM12)

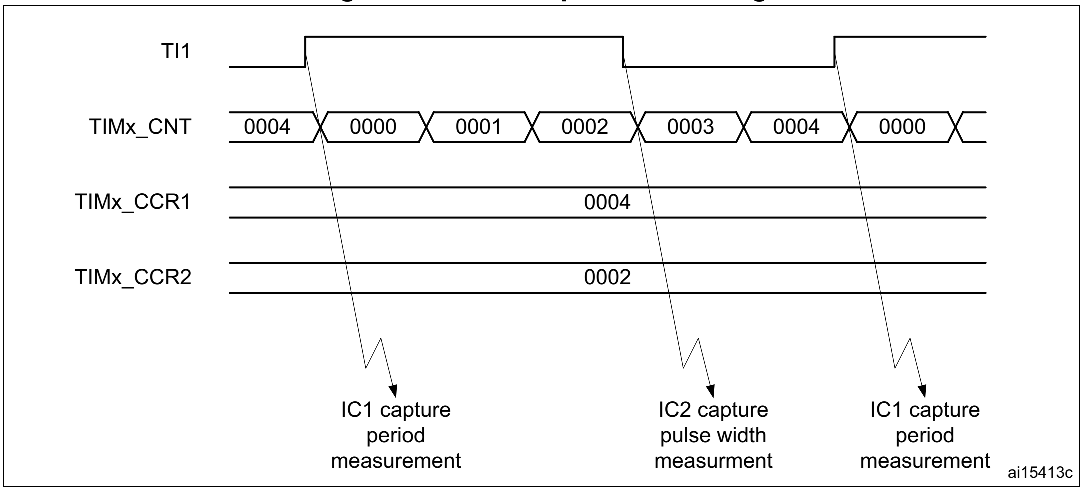
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to ‘01’ in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to ‘00’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to ‘10’ in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to ‘00101’ in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to ‘100’ in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 438. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

40.3.7 Forced output mode

In output mode (CCxS bits = ‘00’ in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write ‘0101’ in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=’0’ (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to ‘0100’ in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

40.3.8 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM='0000'), be set active (OCxM='0001'), be set inactive (OCxM='0010') or can toggle (OCxM='0011') on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

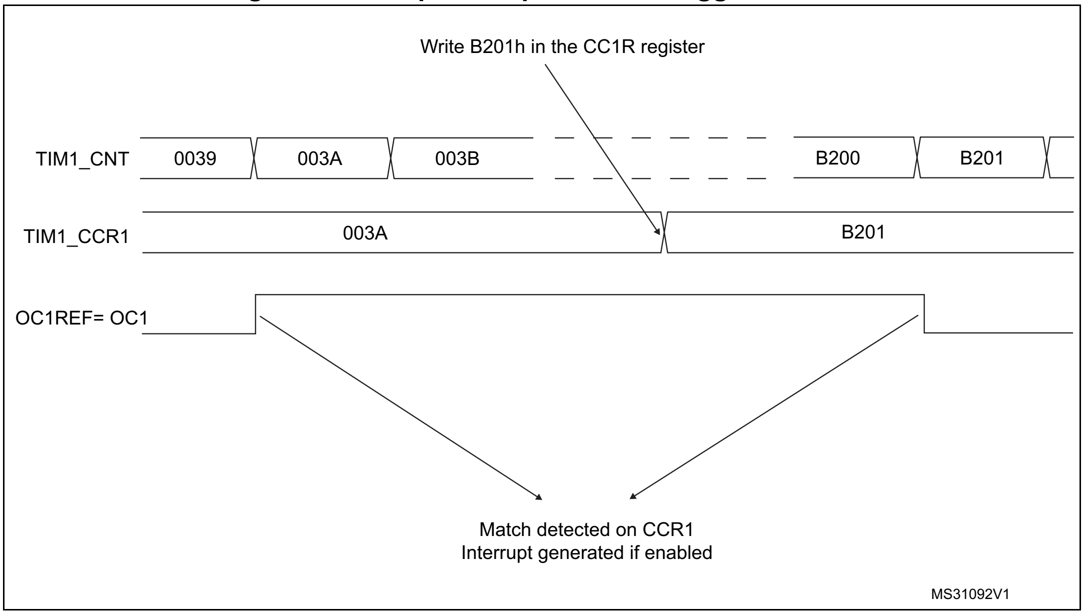
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = '0011' to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = '0' to disable preload register
   - Write CCxP = '0' to select active high polarity
   - Write CCxE = '1' to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 439*.

Figure 439. Output compare mode, toggle on OC1.



40.3.9 PWM mode

Pulse Width Modulation mode allows to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '0110' (PWM mode 1) or '0111' (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

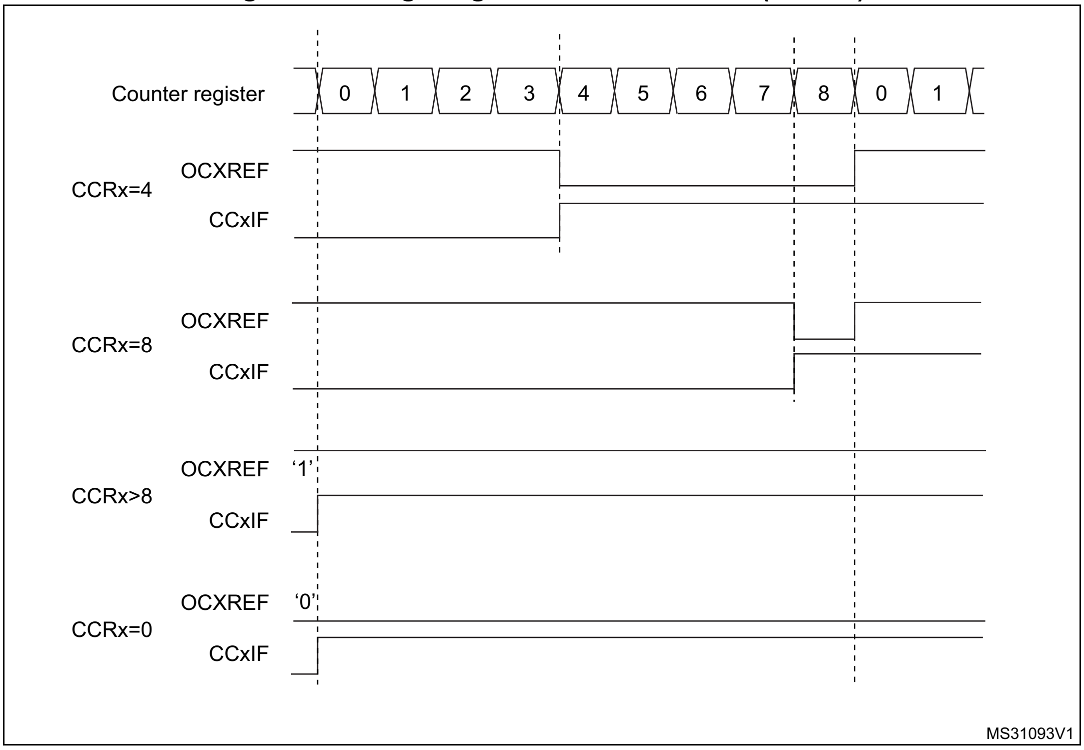
The OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 440* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 440. Edge-aligned PWM waveforms (ARR=8)



40.3.10 Combined PWM mode (TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

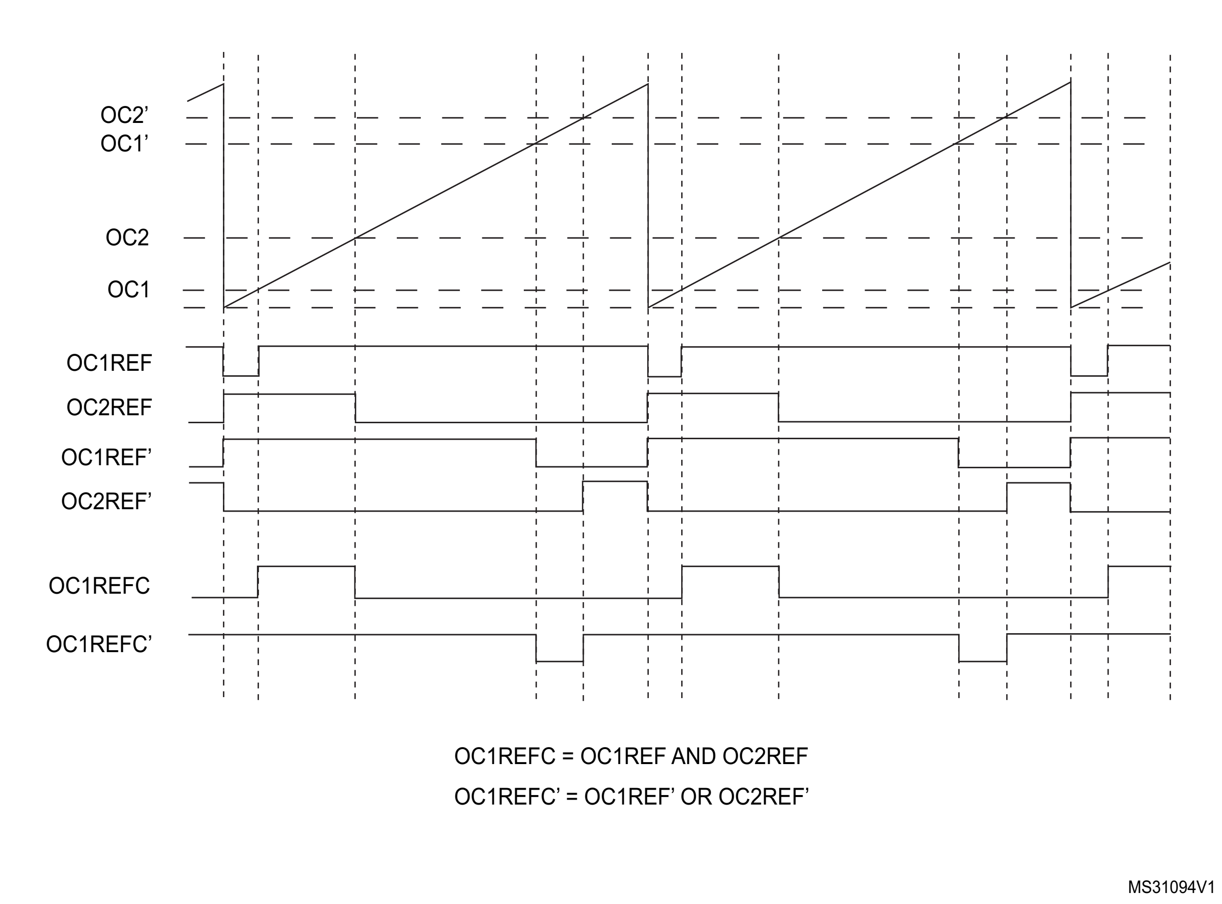
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 441 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 441. Combined PWM mode on channel 1 and 2



40.3.11 One-pulse mode

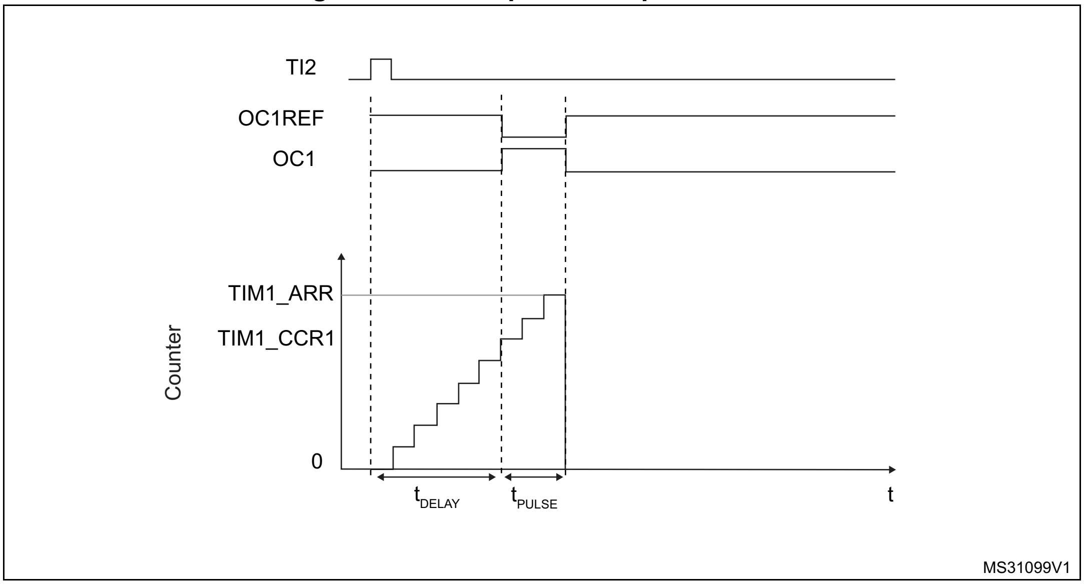
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 442. Example of one pulse mode.



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from '0' to '1' when a compare match occurs and a transition from '1' to '0' when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M='0111' in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE='1' in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to '0' in this example.

Since only 1 pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the TIMx_CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

40.3.12 Retriggerable one pulse mode (TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 40.3.11: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

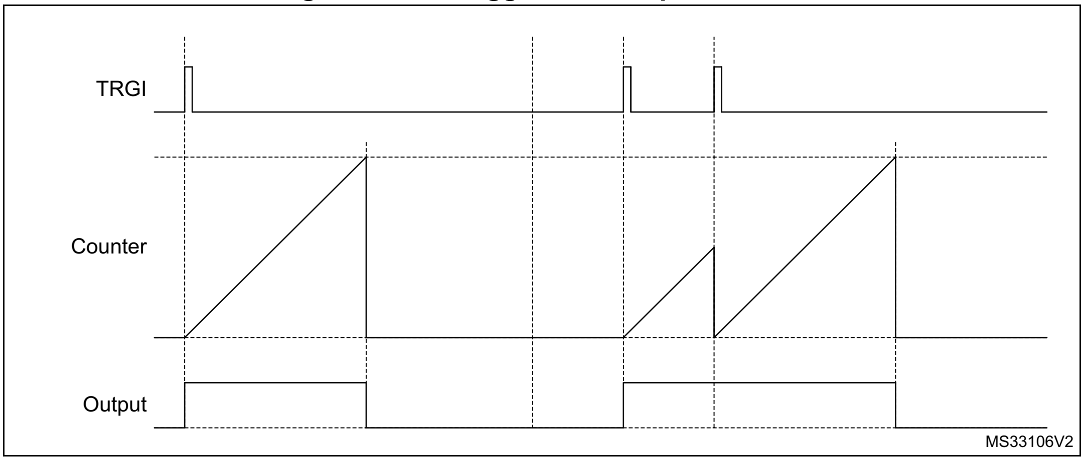
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for retrigerrable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 443. Retriggerable one pulse mode



40.3.13 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the

UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

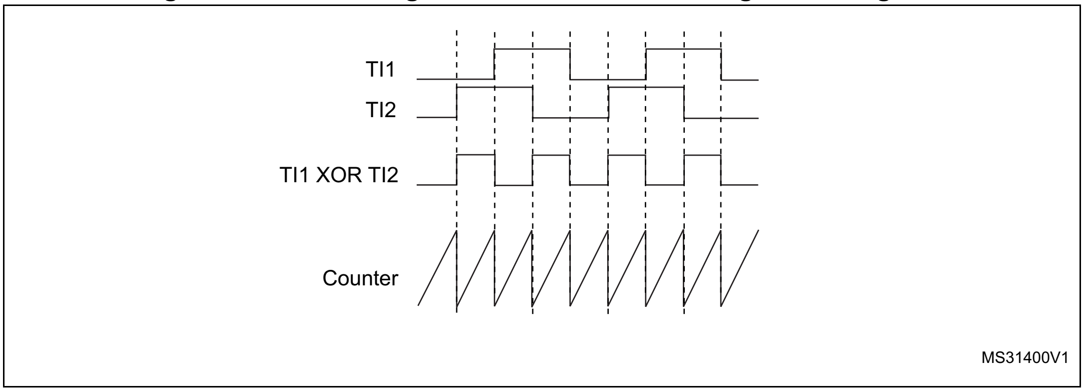
There is no latency between the assertions of the UIF and UIFCPY flags.

40.3.14 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 444*.

Figure 444. Measuring time interval between edges on 2 signals



40.3.15 TIM12 external trigger synchronization

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

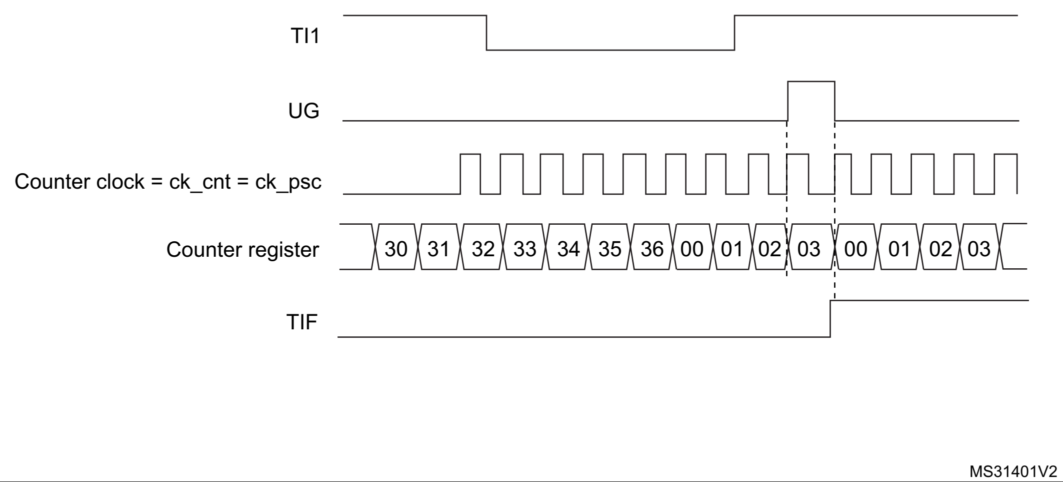
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = '01' in the TIMx_CCMR1 register. Program CC1P and CC1NP to '00' in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS='100' in TIMx_SMCR register. Select TI1 as the input source by writing TS='00101' in TIMx_SMCR register.
3. Start the counter by writing CEN='1' in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 445. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

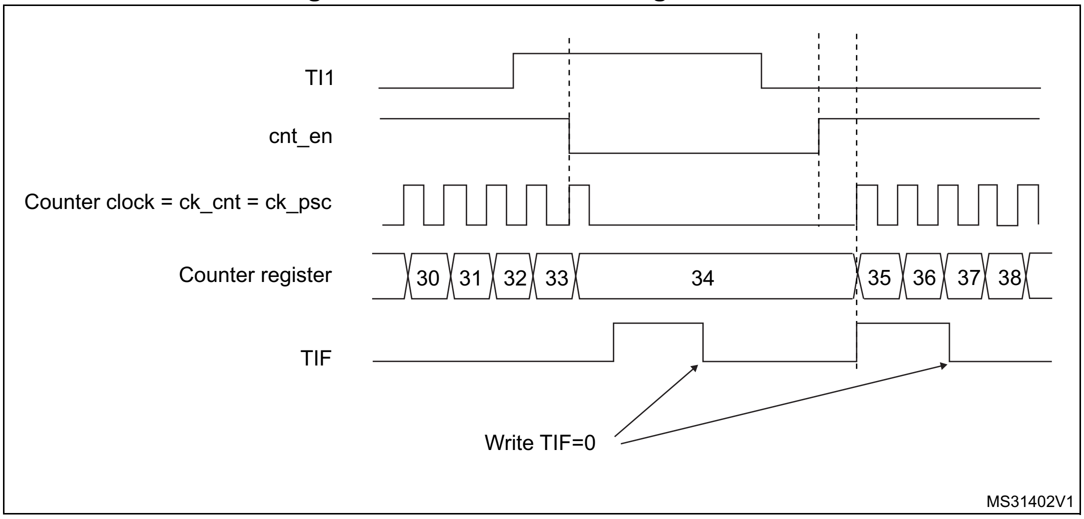
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S='01' in TIMx_CCMR1 register. Program CC1P='1' and CC1NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS='101' in TIMx_SMCR register. Select TI1 as the input source by writing TS='00101' in TIMx_SMCR register.
3. Enable the counter by writing CEN='1' in the TIMx_CR1 register (in gated mode, the counter doesn't start if CEN='0', whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 446. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

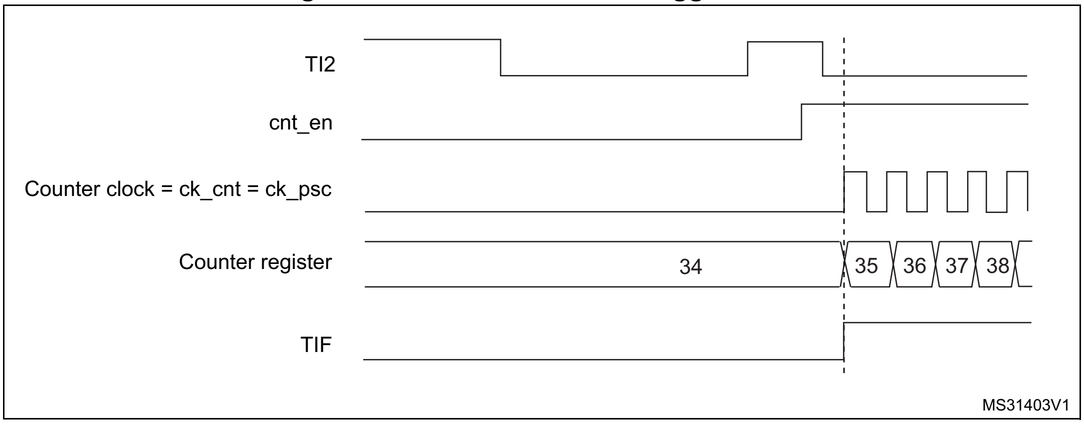
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F='0000'). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S='01' in TIMx_CCMR1 register. Program CC2P='1' and CC2NP='0' in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS='110' in TIMx_SMCR register. Select TI2 as the input source by writing TS='00110' in TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 447. Control circuit in trigger mode



40.3.16 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

40.3.17 Timer synchronization (TIM12)

The TIM timers are linked together internally for timer synchronization or chaining. Refer to *Section 39.3.19: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

40.3.18 Using timer output as trigger for other timers (TIM13/TIM14)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer will detect the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

40.3.19 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on DBG_TIMx_STOP configuration bit in DBGMCU module. For more details, refer to *Section 64.10.9: Microprocessor debug unit (DBGMCU)*

40.4 TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

40.4.1 TIM12 control register 1 (TIM12_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),

00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt if enabled. These events can be:
- Counter overflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.

0: UEV enabled. An UEV is generated by one of the following events:
- Counter overflow
- Setting the UG bit

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

40.4.2 TIM12 control register 2 (TIM12_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: TI1 selection
0: The TIM12_CH1 pin is connected to TI1 input
1: The TIM12_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:
000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.
001: Enable - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
010: Update - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.
011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).
100: Compare - OC1REFC signal is used as trigger output (TRGO).
101: Compare - OC2REFC signal is used as trigger output (TRGO).

Bits 3:0 Reserved, must be kept at reset value.

40.4.3 TIM12 slave mode control register (TIM12_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode
0: No action
1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.
00000: Internal Trigger 0 (ITR0)
00001: Internal Trigger 1 (ITR1)
00010: Internal Trigger 2 (ITR2)
00011: Internal Trigger 3 (ITR3)
00100: TI1 Edge Detector (TI1F_ED)
00101: Filtered Timer Input 1 (TI1FP1)
00110: Filtered Timer Input 2 (TI2FP2)
Others: Reserved
See *Table 286: TIMx internal trigger connection on page 2107* for more details on the meaning of ITRx for each timer.

Note: These bits must be changed only when they are not used (e.g. when SMS='000') to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).
0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='00100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Table 286. TIMx internal trigger connection

Slave TIM	ITR0 (TS = '00000')	ITR1 (TS = '00001')	ITR2 (TS = '00010')	ITR3 (TS = '00011')
TIM12	TIM4	TIM5	TIM13 OC1	TIM14 OC1

40.4.4 TIM12 Interrupt enable register (TIM12_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

40.4.5 TIM12 status register (TIM12_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow and if UDIS='0' in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS='0' and UDIS='0' in the TIMx_CR1 register.
- When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS='0' and UDIS='0' in the TIMx_CR1 register.

40.4.6 TIM12 event generation register (TIM12_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

40.4.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

40.4.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas the active level of OC1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The OC1REF signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. The OC1REF signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1

0100: Force inactive level - OC1REF is forced low

0101: Force active level - OC1REF is forced high

0110: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved,

1111: Reserved

Note: In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

40.4.9 TIM12 capture/compare enable register (TIM12_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity

Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity

Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable

Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity

CC1 channel configured as output: CC1NP must be kept cleared

CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 287. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)’

Note: The states of the external I/O pins connected to the standard OCx channels depend on the state of the OCx channel and on the GPIO registers.

40.4.10 TIM12 counter (TIM12_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

40.4.11 TIM12 prescaler (TIM12_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event.
(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

40.4.12 TIM12 auto-reload register (TIM12_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.
Refer to the *Section 40.3.1: Time-base unit on page 2082* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

40.4.13 TIM12 capture/compare register 1 (TIM12_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded into the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit). Else the preload value is copied into the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signaled on the OC1 output.
If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1).

40.4.14 TIM12 capture/compare register 2 (TIM12_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:
CCR2 is the value to be loaded into the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (OC2PE bit). Else the preload value is copied into the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the TIMx_CNT counter and signalled on the OC2 output.
If channel CC2 is configured as input:
CCR2 is the counter value transferred by the last input capture 2 event (IC2).

40.4.15 TIM12 timer input selection register (TIM12_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input

0000: TIM12_CH2 input
Other: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM12_CH1 input
0001: hsi_cal_ck
0010: csi_cal_ck
Other: Reserved

40.4.16 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 288. TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIM12_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x08	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3] Res.		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x24	**TIMx_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 288. TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**TIMx_ARR**	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved	Reserved																															
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x64	Reserved	Res.																															
0x68	**TIM12_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

40.5 TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits) or words (32 bits).

40.5.1 TIMx control register 1 (TIMx_CR1)(x = 13 to 14)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and sampling clock used by the digital filters (TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 \times t_{CK\_INT}$
10: $t_{DTS} = 4 \times t_{CK\_INT}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

– Counter overflow

– Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

– Counter overflow

– Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

40.5.2 TIMx Interrupt enable register (TIMx_DIER)(x = 13 to 14)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

40.5.3 TIMx status register (TIMx_SR)(x = 13 to 14)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow and if UDIS=’0’ in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=’0’ and UDIS=’0’ in the TIMx_CR1 register.

40.5.4 TIMx event generation register (TIMx_EGR)(x = 13 to 14)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

40.5.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13 to 14)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

40.5.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the

corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal OC1REF from which OC1 is derived. OC1REF is active high whereas OC1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

Note: In PWM mode 1 or 2, the OCREF level changes when the result of the comparison changes or when the output compare mode switches from frozen to PWM mode.

Note: The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: Reserved.

11: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

40.5.7 TIMx capture/compare enable register (TIMx_CCER)(x = 13 to 14)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0:This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 289. Output control bit for standard OCx channels

CCxE bit	OCx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external I/O pins connected to the standard OCx channels depends on the OCx channel state and the GPIO registers.

40.5.8 TIMx counter (TIMx_CNT)(x = 13 to 14)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

40.5.9 TIMx prescaler (TIMx_PSC)(x = 13 to 14)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

40.5.10 TIMx auto-reload register (TIMx_ARR)(x = 13 to 14)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to *Section 40.3.1: Time-base unit on page 2082* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

40.5.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13 to 14)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (IC1).

40.5.12 TIM13 timer input selection register (TIM13_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM13_CH1 input
Other: Reserved

40.5.13 TIM14 timer input selection register (TIM14_TISEL)

Address offset: 0x68

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM14_CH1 input
Other: Reserved

40.5.14 TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 290. TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04 to 0x08	Reserved	Res.																															
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0								0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x18	TIMx_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0
0x24	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved	Res.																															

Table 290. TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x64	Reserved	Res.																															
0x68	**TIM13_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x68	**TIM14_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

41 General-purpose timers (TIM15/TIM16/TIM17)

41.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 41.4.23: Timer synchronization (TIM15)*.

41.2 TIM15 main features

TIM15 includes the following features:

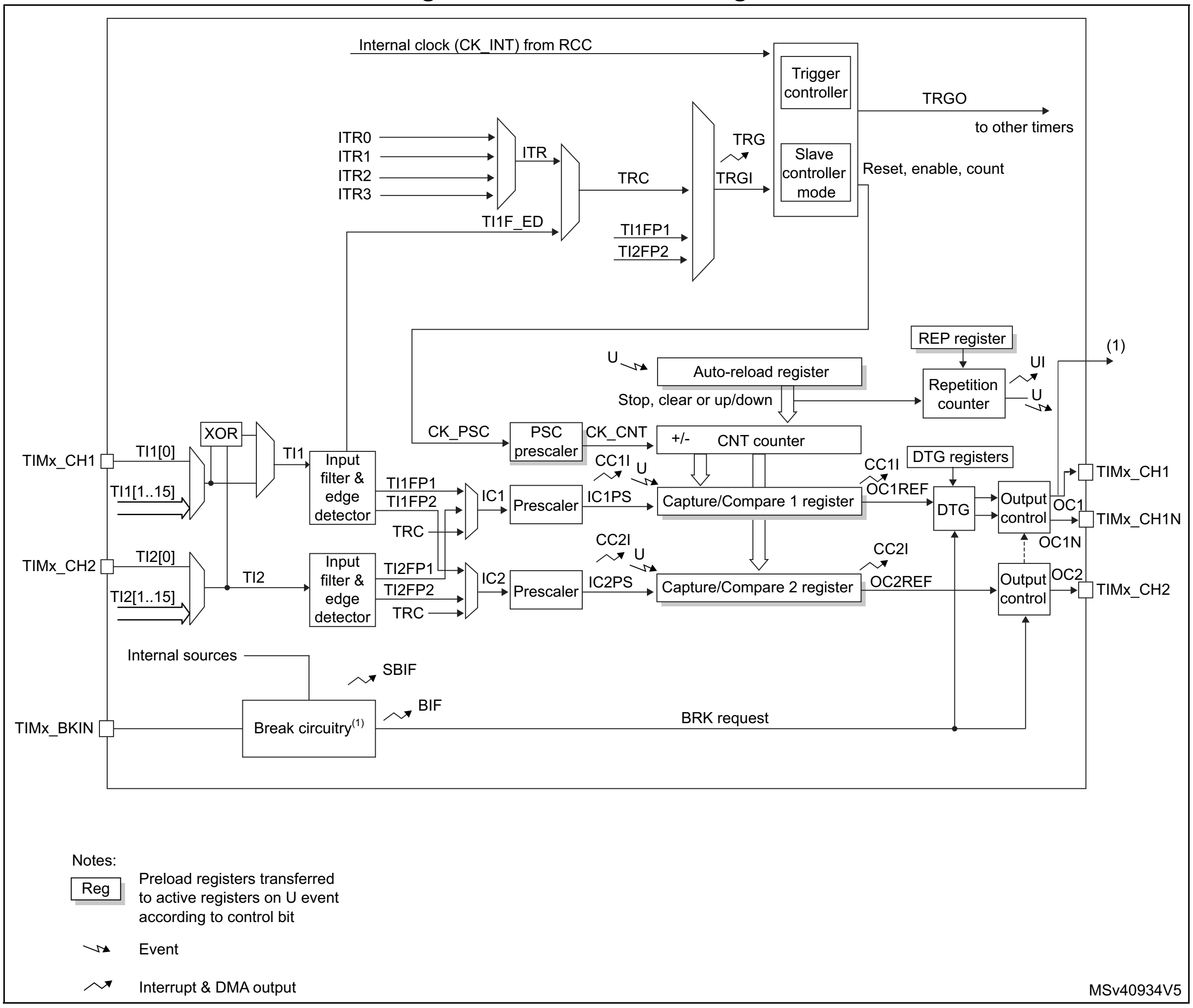
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to 2 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

41.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

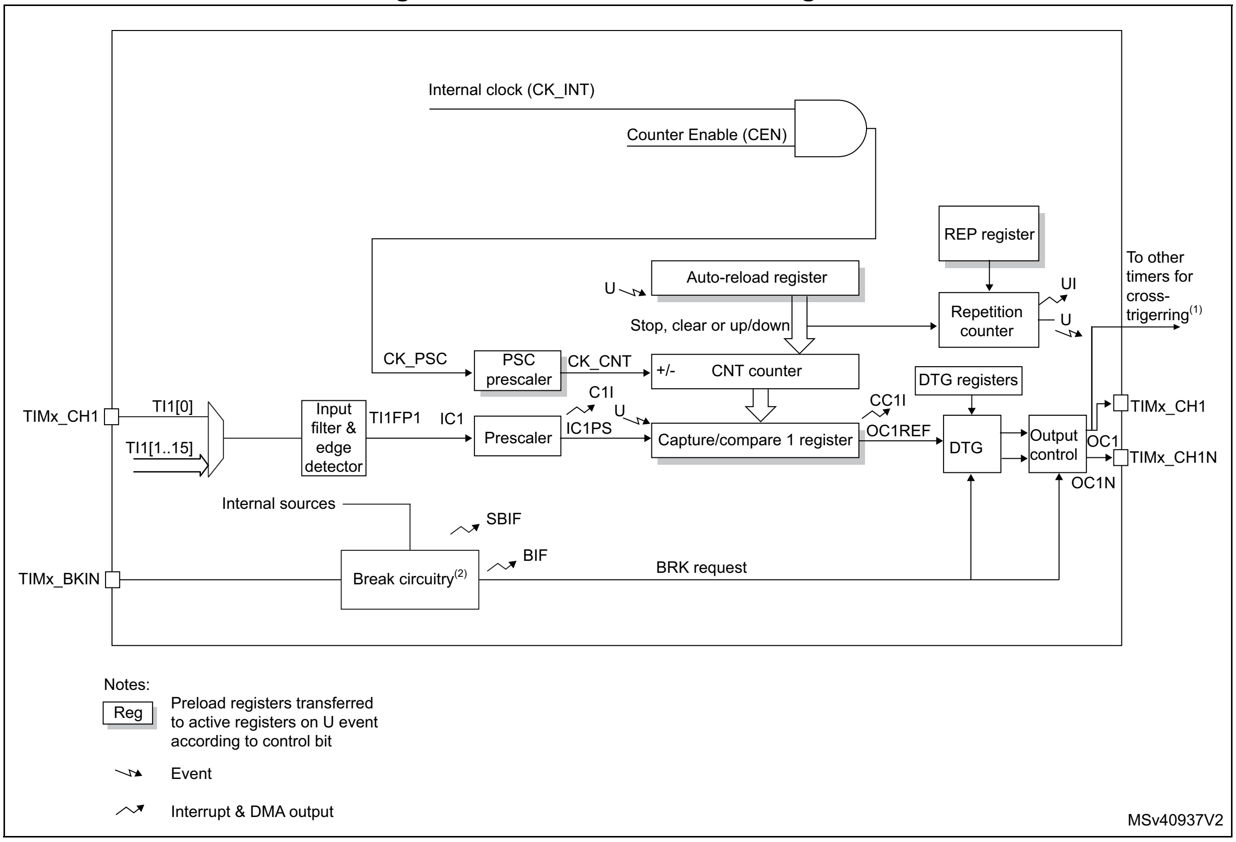
- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

Figure 448. TIM15 block diagram



1. The internal break event source can be:
   - A PVD output
   - Cortex®-M4 with FPU LOCKUP (Hardfault) output

Figure 449. TIM16/TIM17 block diagram



1. This signal can be used as trigger for some slave timer, see *Section 41.4.24: Using timer output as trigger for other timers (TIM16/TIM17)*.
2. The internal break event source can be:
   - A PVD output
   - Cortex®-M4 with FPU LOCKUP (Hardfault) output

41.4 TIM15/TIM16/TIM17 functional description

41.4.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 450 and *Figure 451* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 450. Counter timing diagram with prescaler division change from 1 to 2

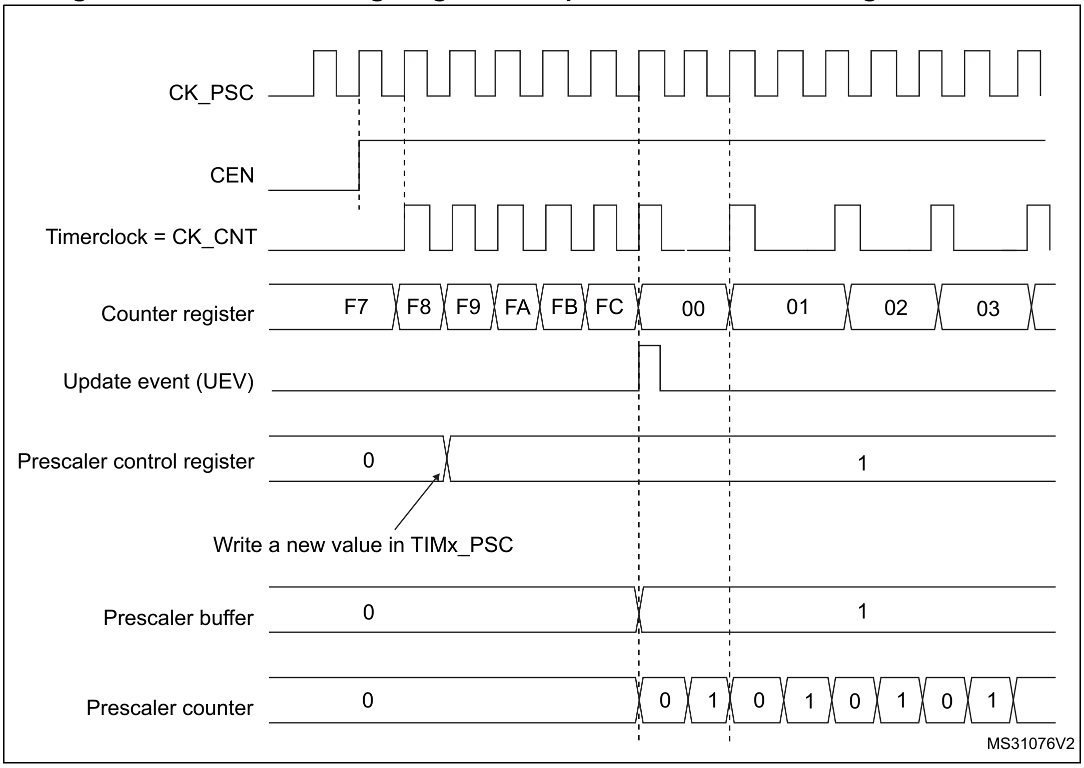
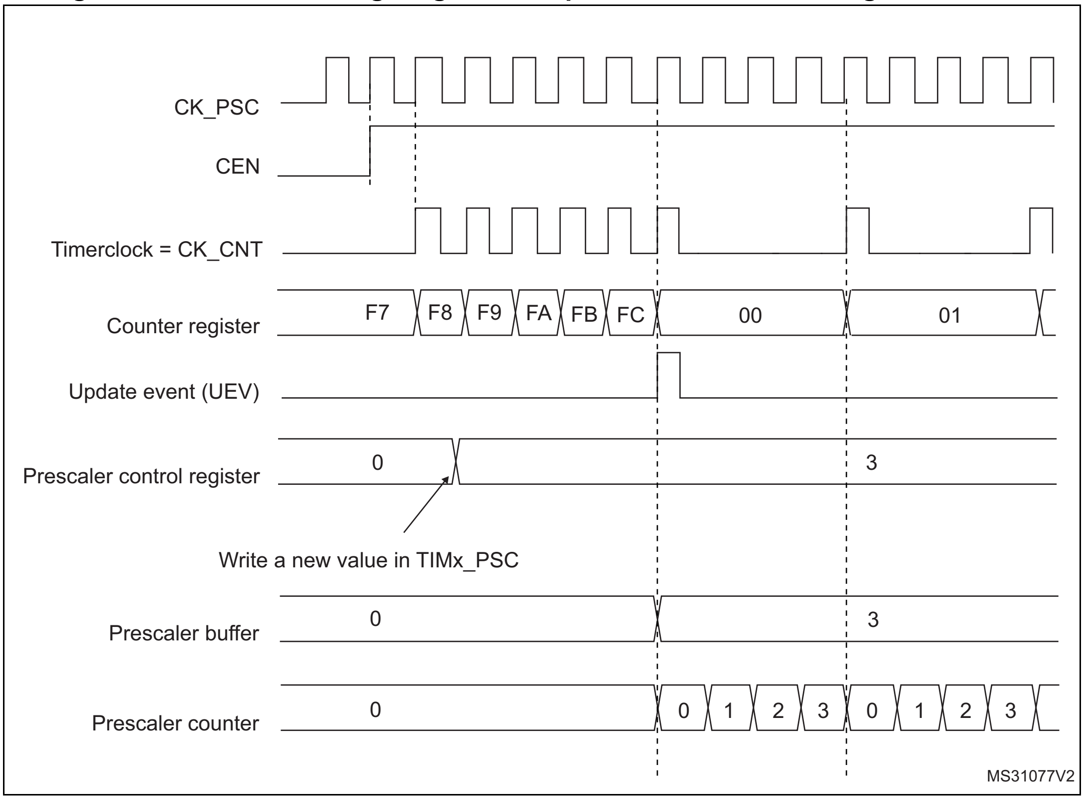


Figure 451. Counter timing diagram with prescaler division change from 1 to 4



41.4.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The auto-reload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR=0x36.

Figure 452. Counter timing diagram, internal clock divided by 1

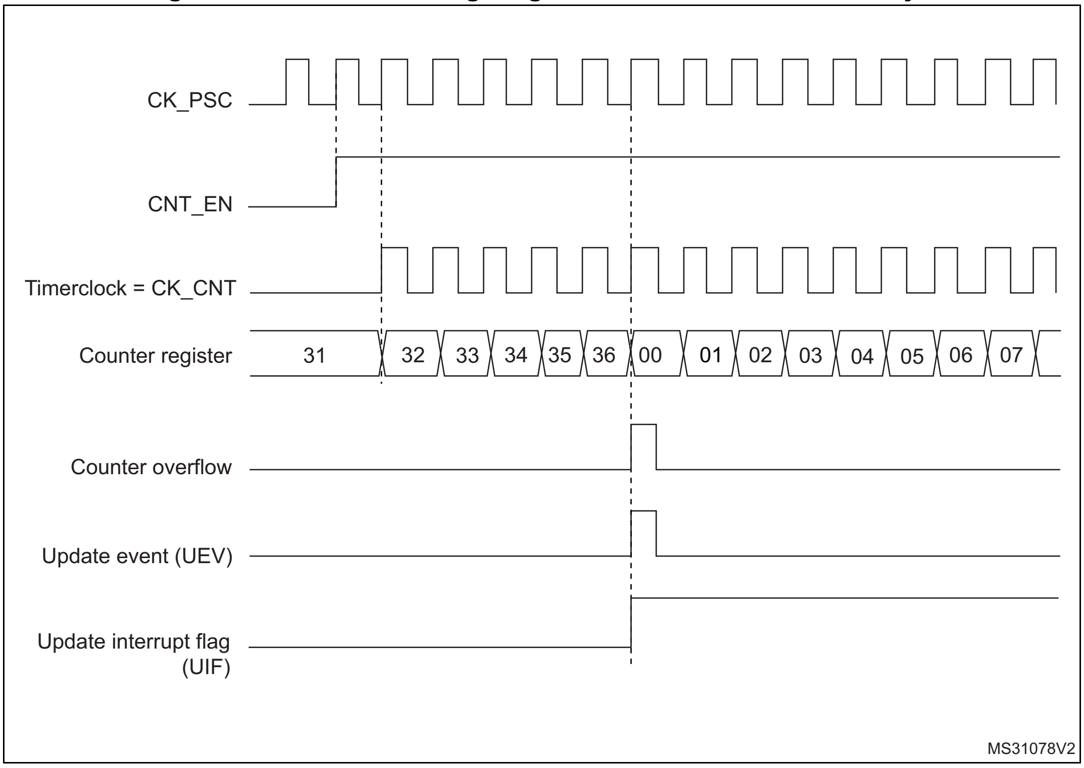


Figure 453. Counter timing diagram, internal clock divided by 2

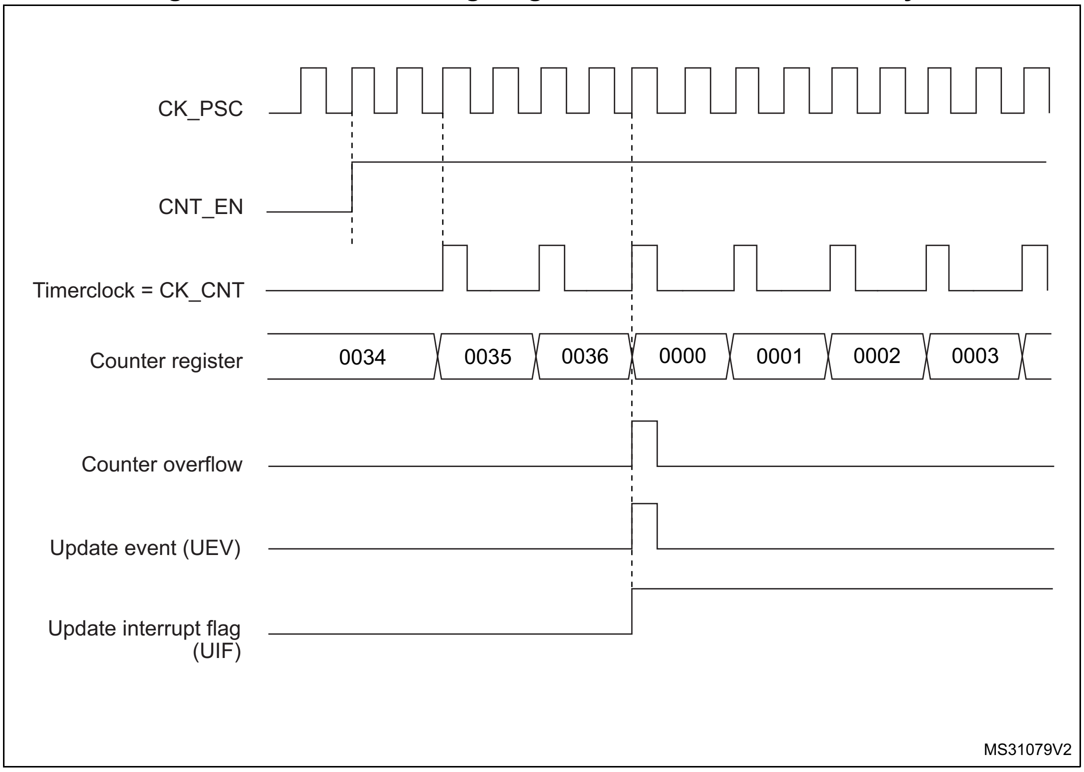


Figure 454. Counter timing diagram, internal clock divided by 4

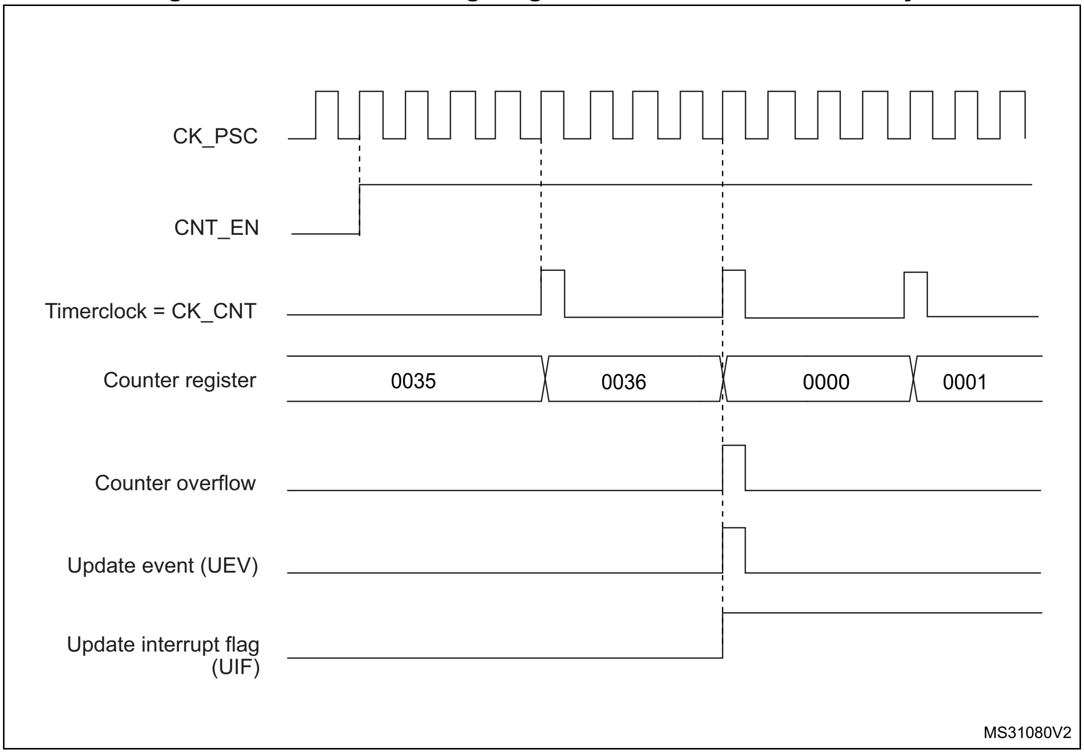


Figure 455. Counter timing diagram, internal clock divided by N

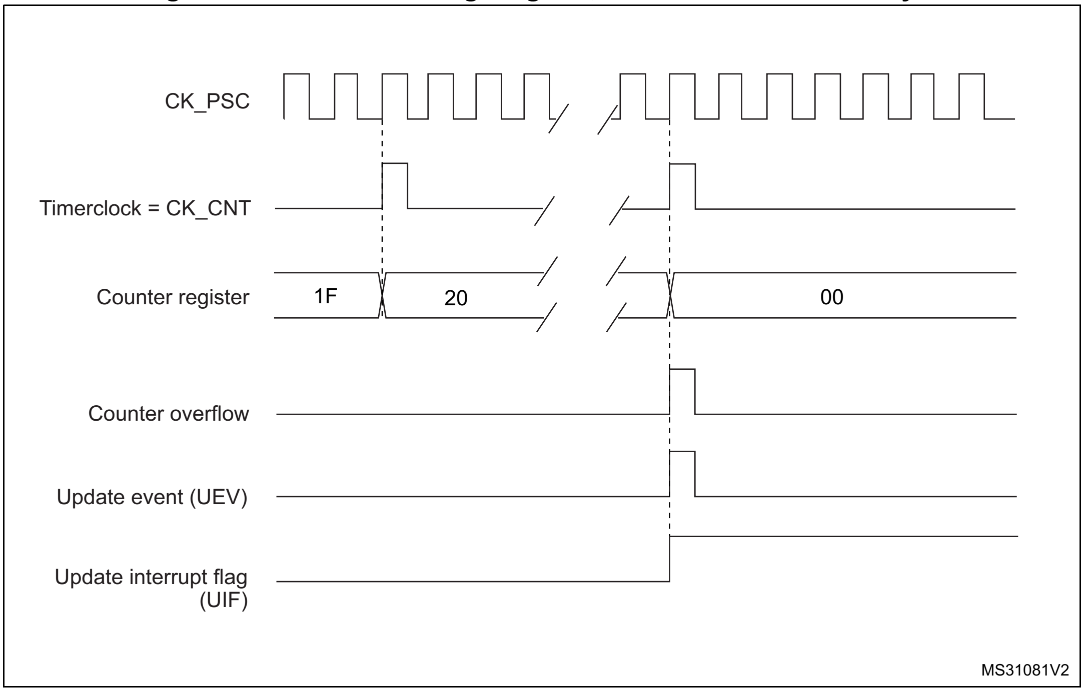


Figure 456. Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

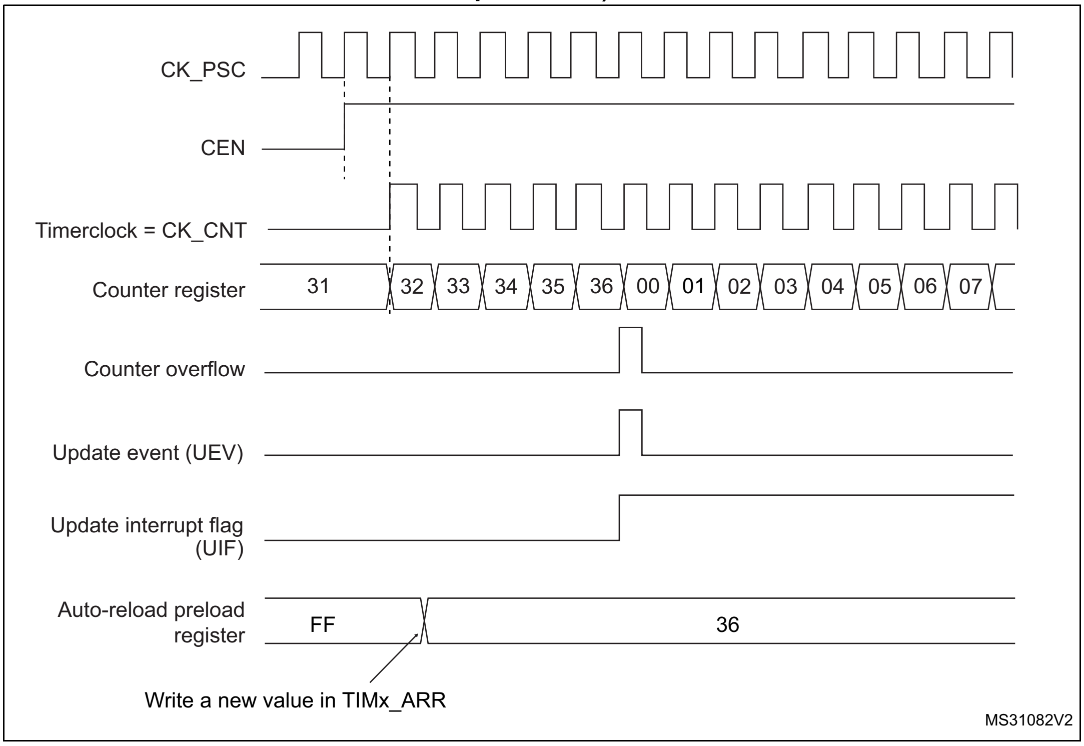
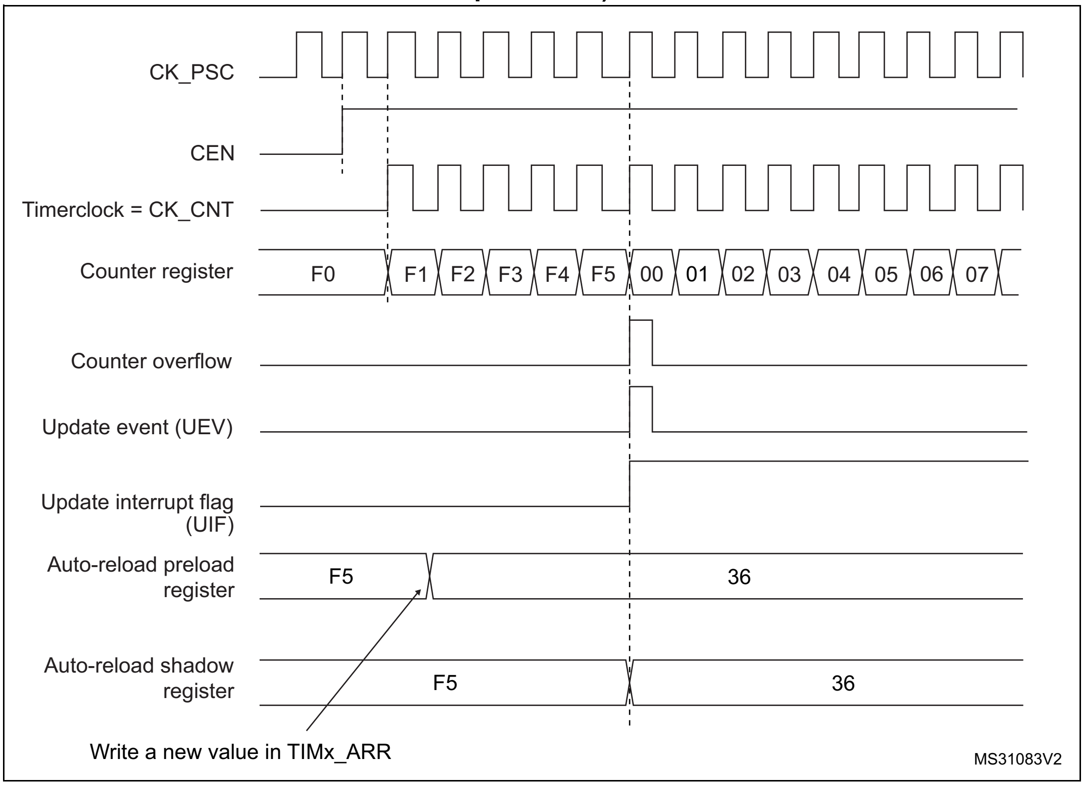


Figure 457. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



41.4.3 Repetition counter

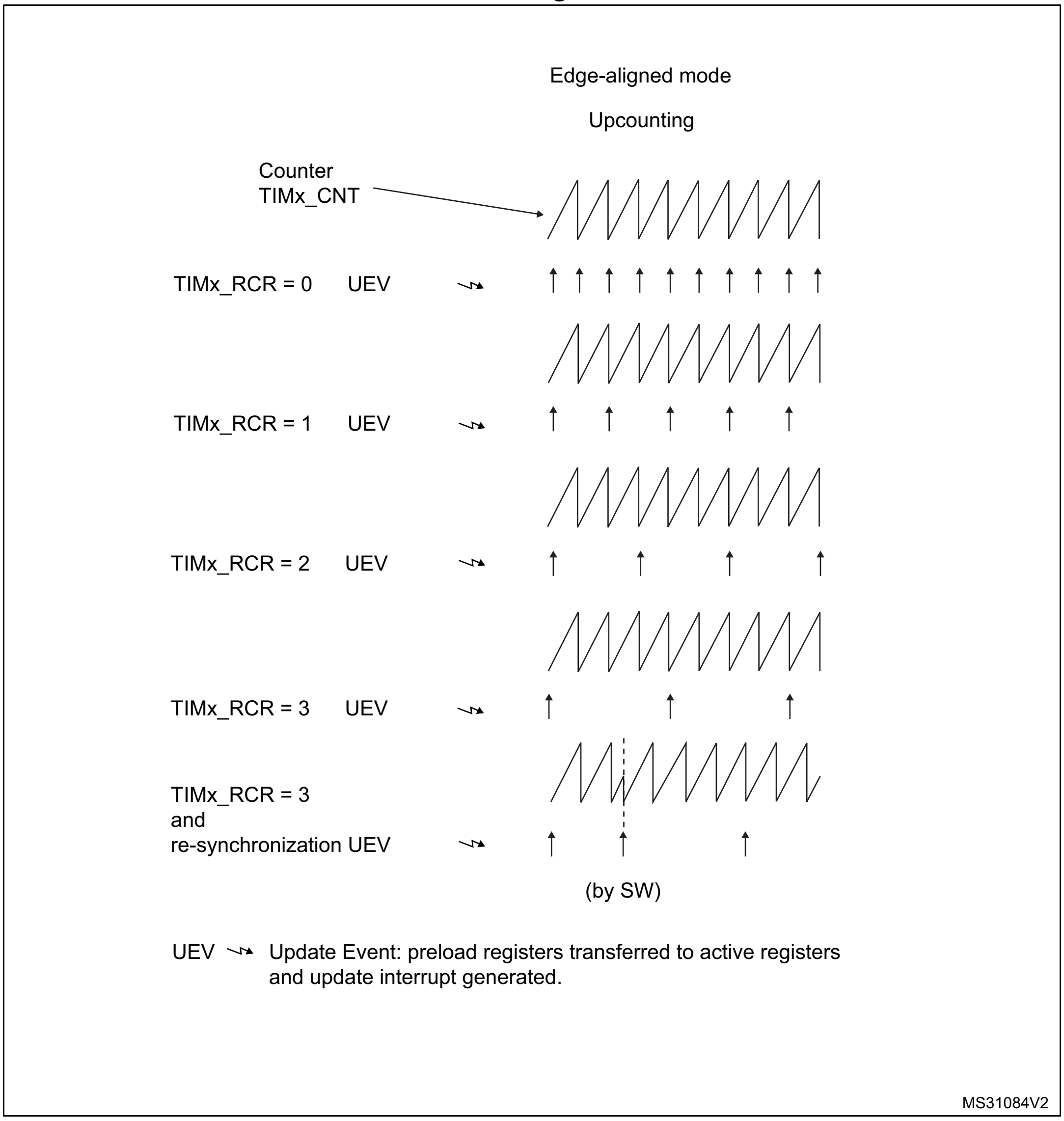
Section 41.4.1: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 458*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 458. Update rate examples depending on mode and TIMx_RCR register settings



41.4.4 Clock selection

The counter clock can be provided by the following clock sources:

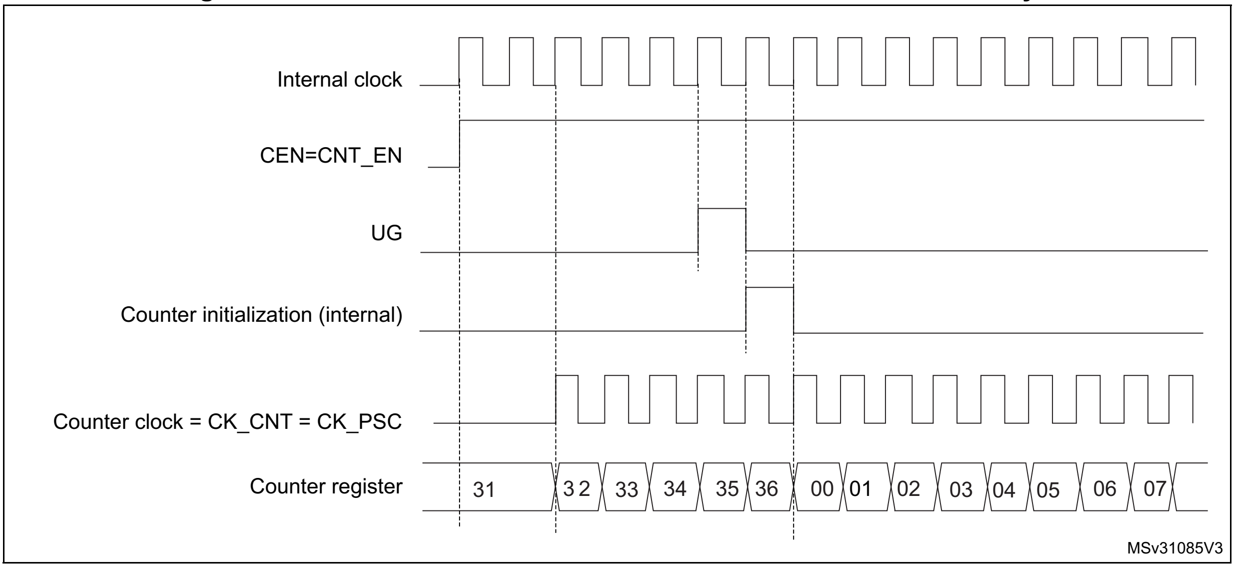
- Internal clock (CK_INT)
- External clock mode1: external input pin
- Internal trigger inputs (ITRx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer on page 2043* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 459 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

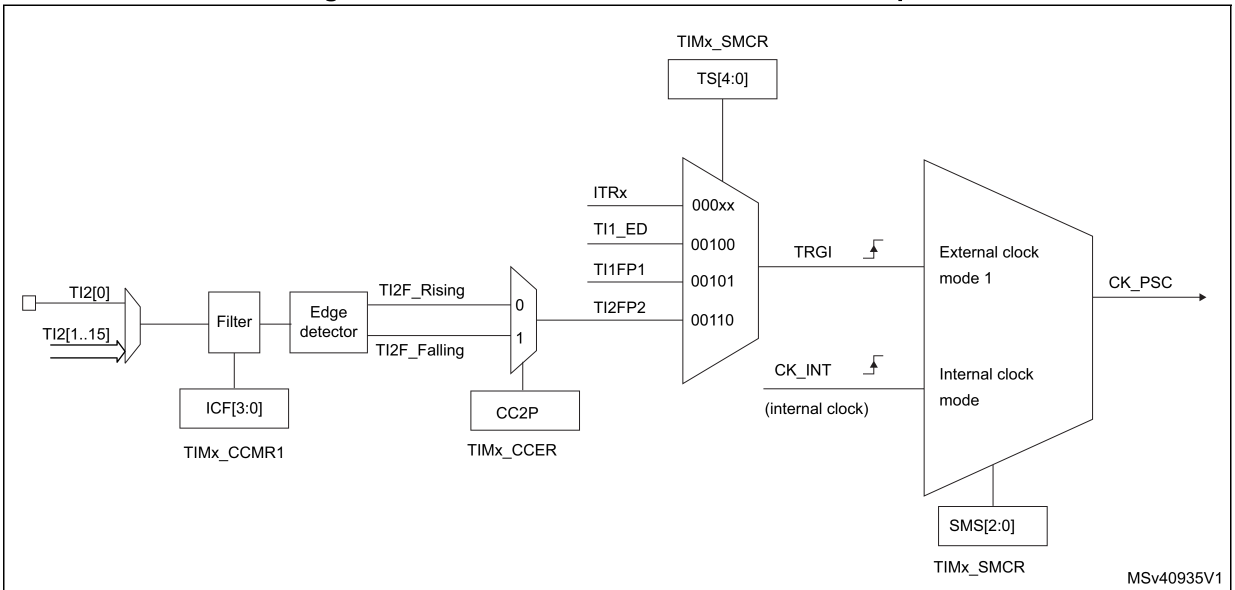
Figure 459. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS=111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 460. TI2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

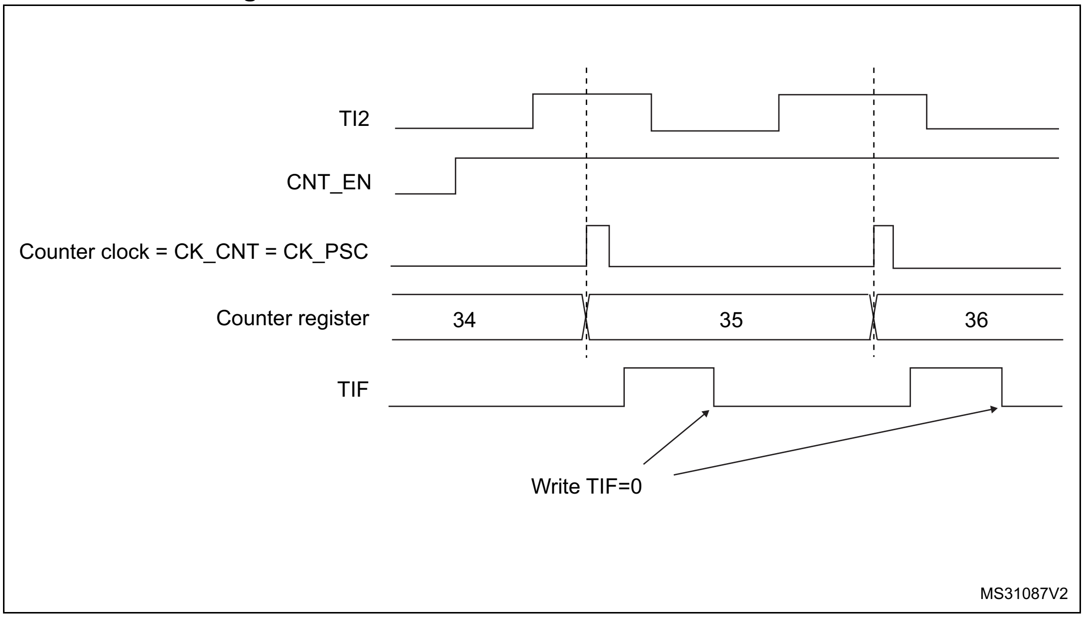
1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F=0000).
4. Select rising edge polarity by writing CC2P=0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS=111 in the TIMx_SMCR register.
6. Select TI2 as the trigger input source by writing TS=00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN=1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, so it does not need to be configured.

When a rising edge occurs on TI2, the counter counts once and the TIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on TI2 input.

Figure 461. Control circuit in external clock mode 1



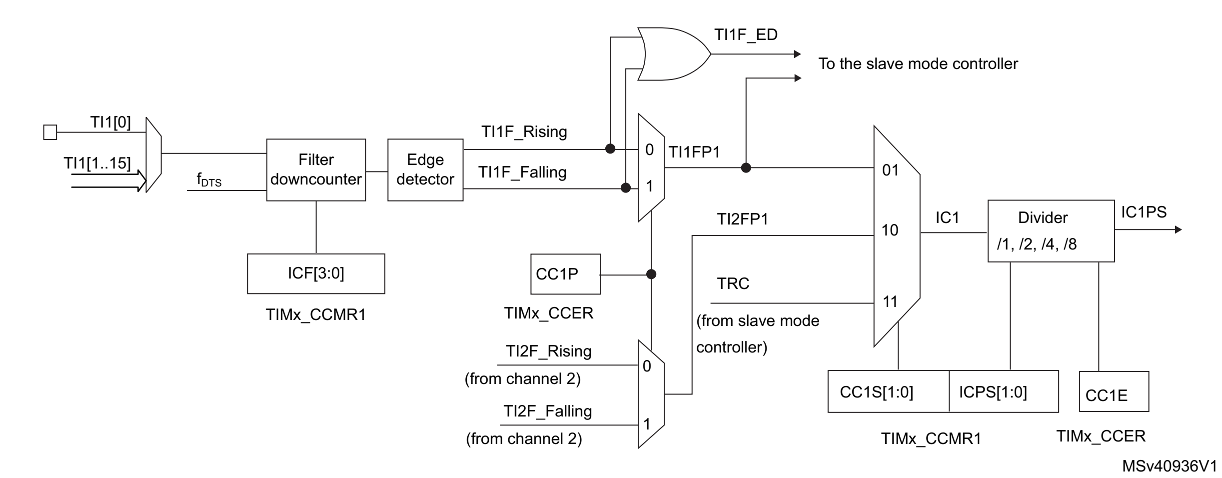
41.4.5 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 462 to *Figure 465* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 462. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 463. Capture/compare channel 1 main circuit

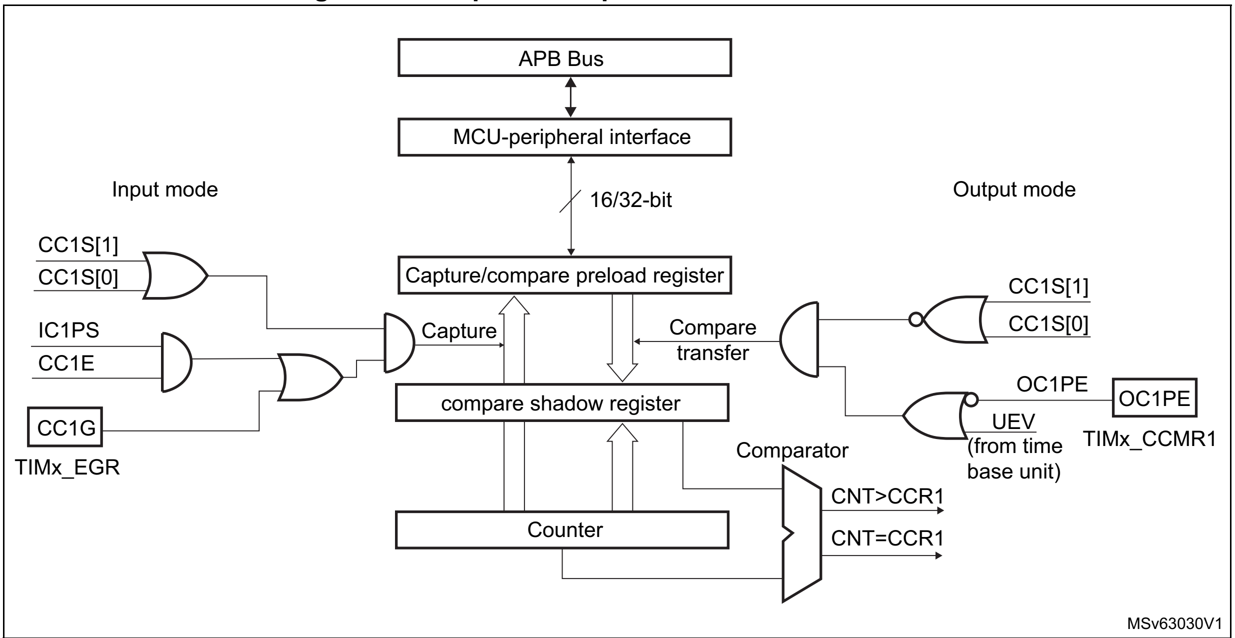


Figure 464. Output stage of capture/compare channel (channel 1)

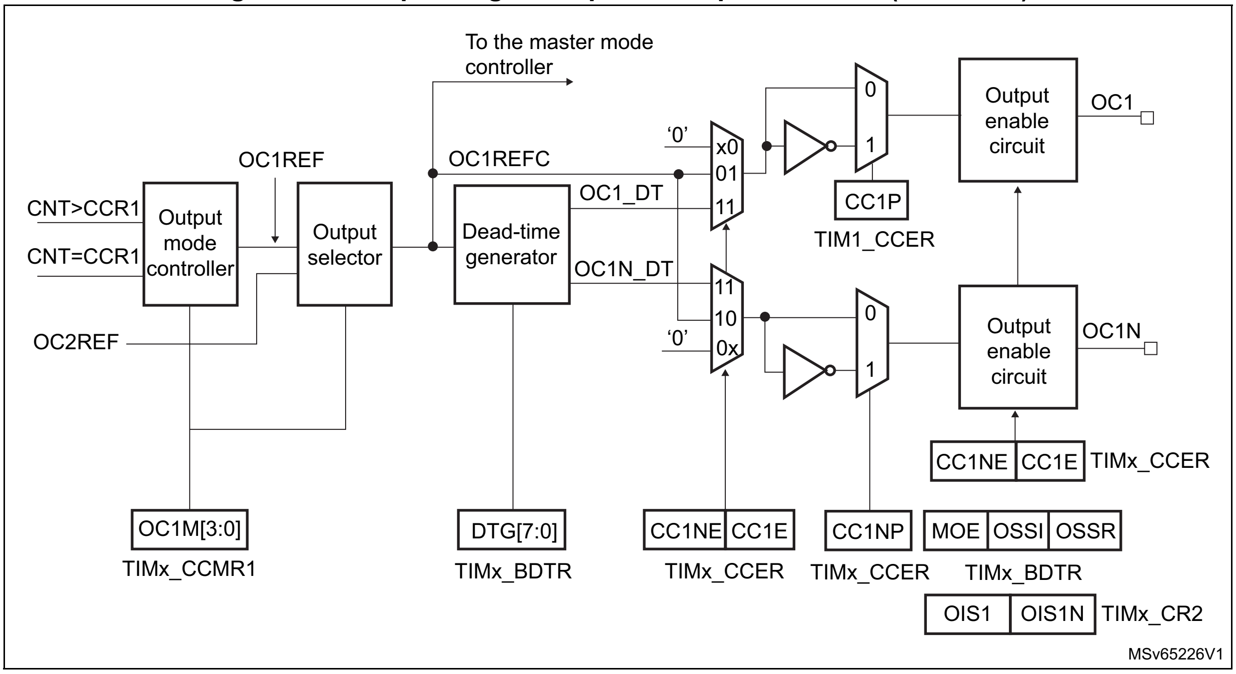
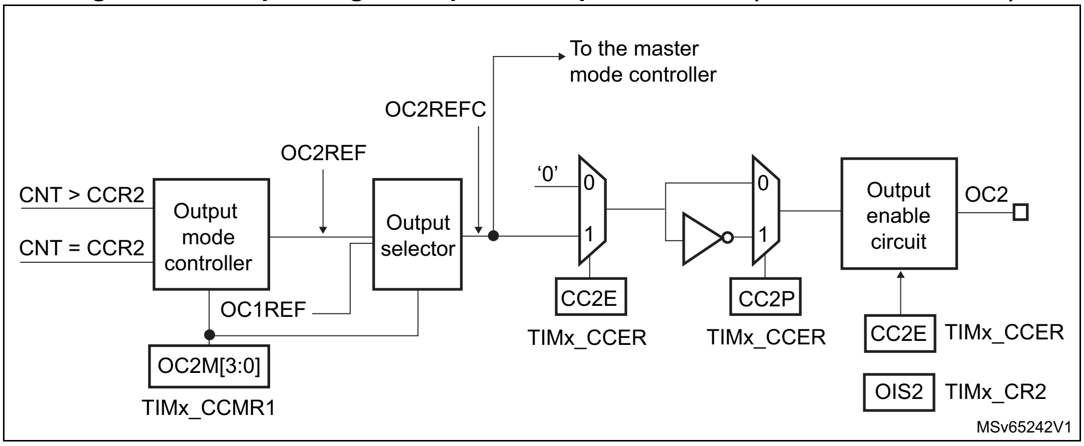


Figure 465. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

41.4.6 Input capture mode

In Input capture mode, the Capture/Compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to ‘0’ or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when TI1 input rises. To do this, use the following procedure:

1. Select the proper TI1x source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the TIx (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. We must program a filter duration longer than these 5 clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to ‘00’ in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

41.4.7 PWM input mode (only for TIM15)

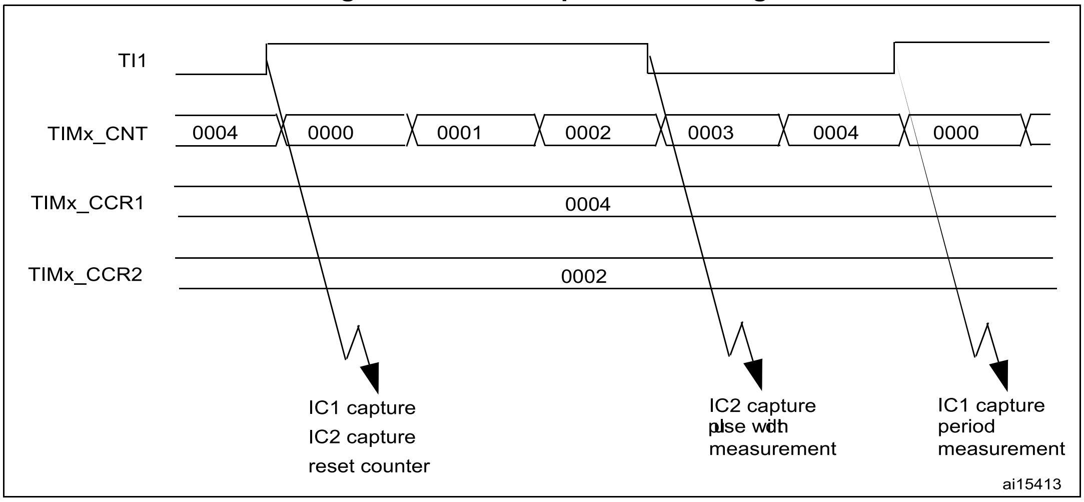
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

1. Select the proper TI1[x] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
3. Select the active polarity for TI1FP1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to ‘0’ (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
5. Select the active polarity for TI1FP2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to ‘10’ (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (TI1FP1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to ‘1’ in the TIMx_CCER register.

Figure 466. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

41.4.8 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

41.4.9 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

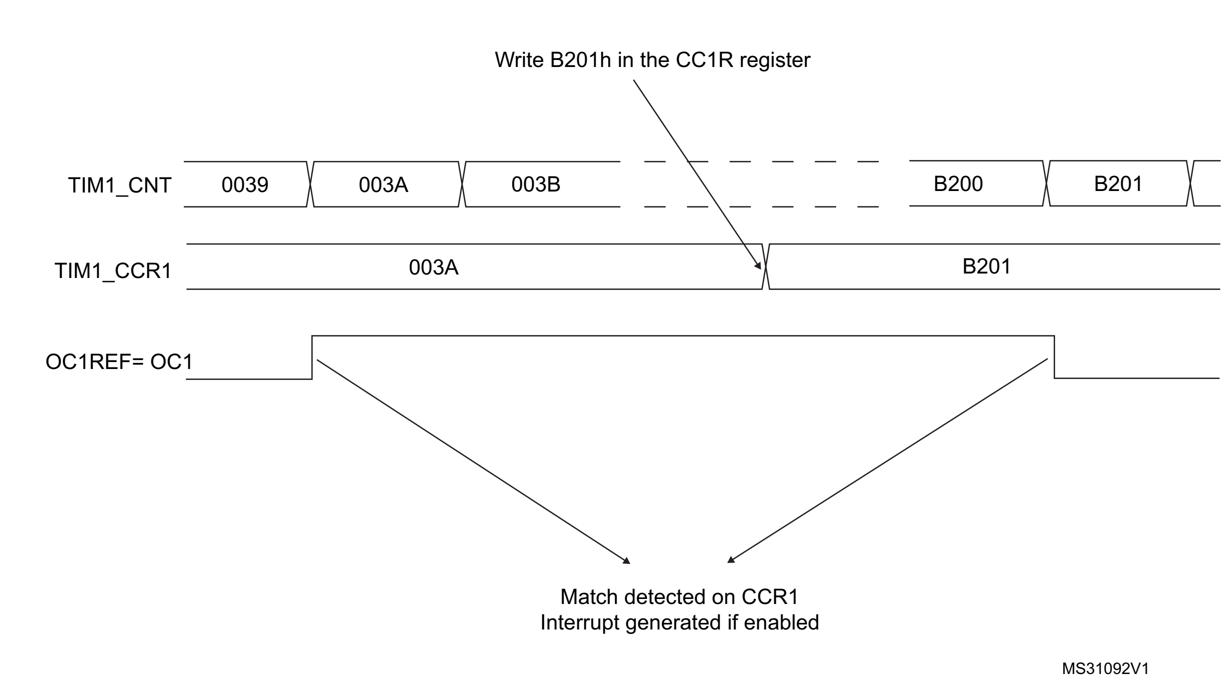
In output compare mode, the update event UEV has no effect on OCxREF and OCx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   – Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   – Write OCxPE = 0 to disable preload register
   – Write CCxP = 0 to select active high polarity
   – Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 467*.

Figure 467. Output compare mode, toggle on OC1



41.4.10 PWM mode

Pulse Width Modulation mode allows a signal to be generated with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing ‘110’ (PWM mode 1) or ‘111’ (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

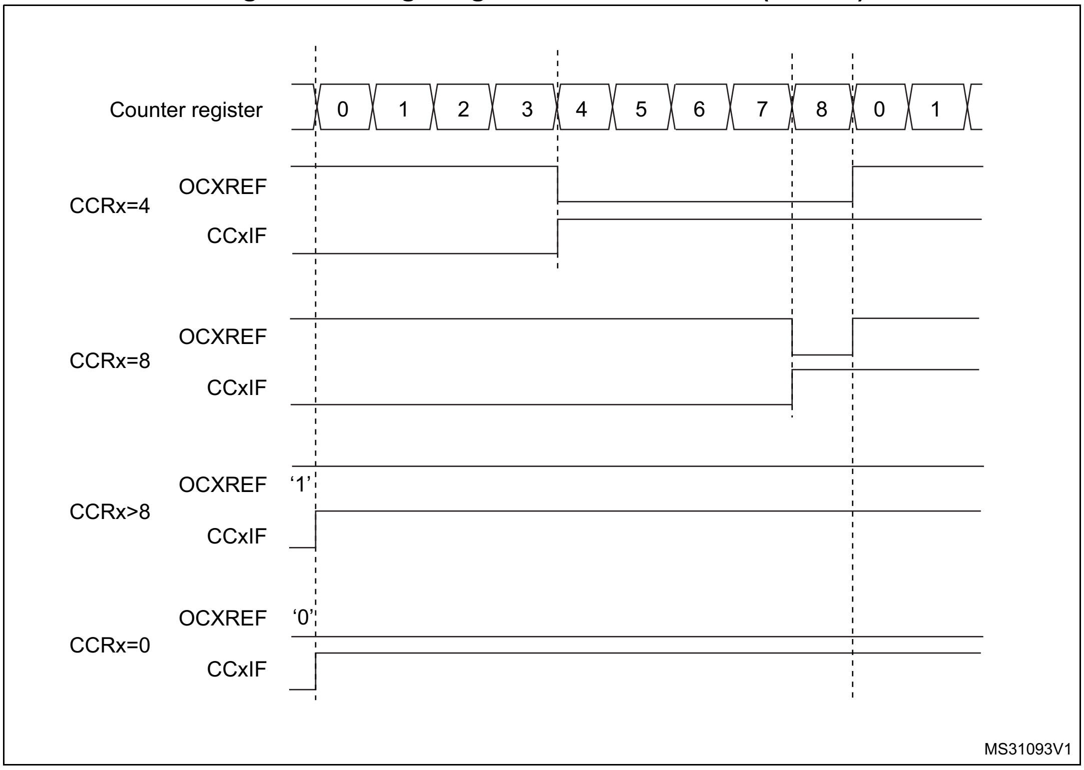
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode on page 2138*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the auto-reload value (in TIMx_ARR) then OCxREF is held at

'1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 468* shows some edge-aligned PWM waveforms in an example where TIMx_ARR=8.

Figure 468. Edge-aligned PWM waveforms (ARR=8)



41.4.11 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, OCxREFC, are made of an OR or AND logical combination of two reference PWMs:

- OC1REFC (or OC2REFC) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers

Combined PWM mode can be selected independently on two channels (one OCx output per pair of CCR registers) by writing '1100' (Combined PWM mode 1) or '1101' (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

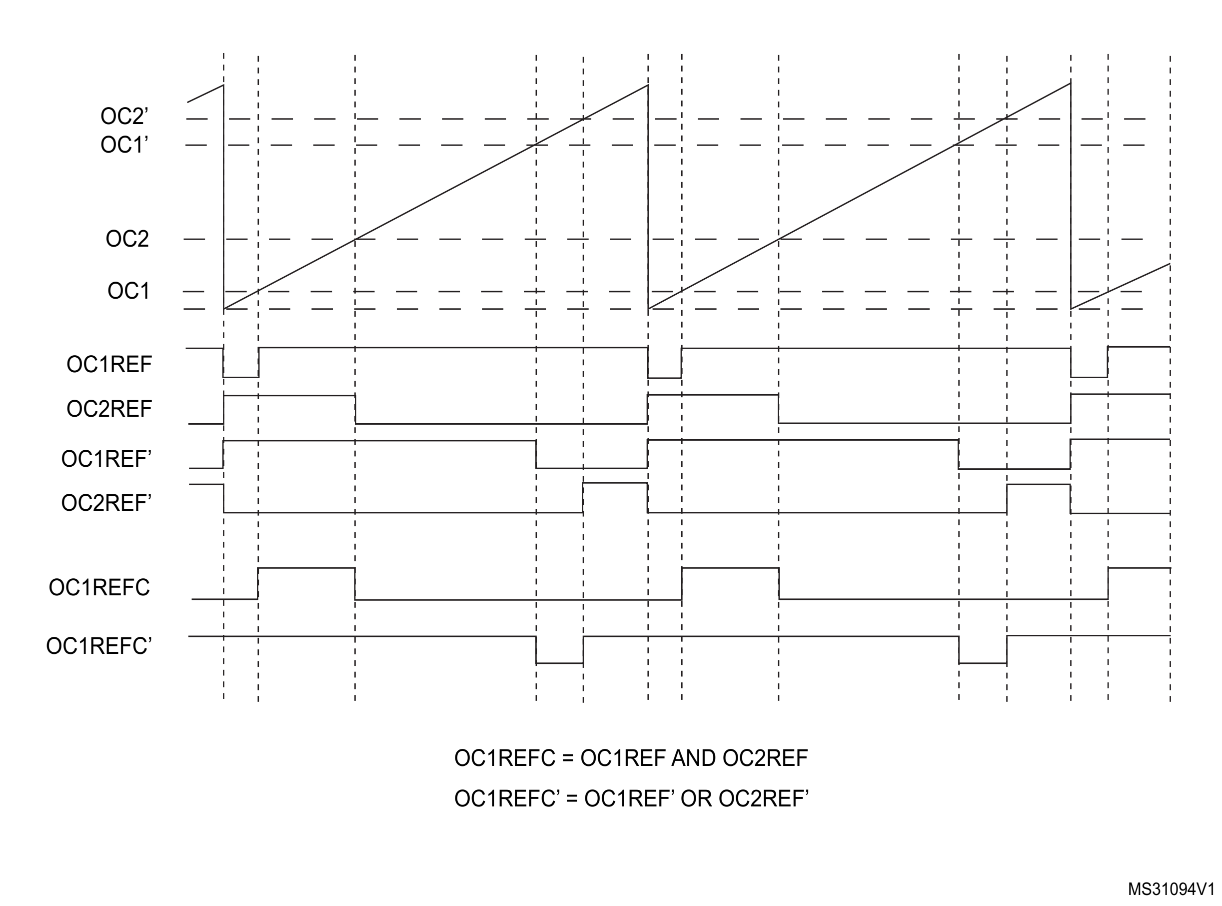
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bit field is split into two parts for compatibility reasons, the most significant bit is not contiguous with the 3 least significant ones.

Figure 469 represents an example of signals that can be generated using Asymmetric PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2,
- Channel 2 is configured in PWM mode 1,

Figure 469. Combined PWM mode on channel 1 and 2



41.4.12 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

The polarity of the outputs (main output OCx or complementary OCxN) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 295: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17) on page 2207* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 470. Complementary output with dead-time insertion.

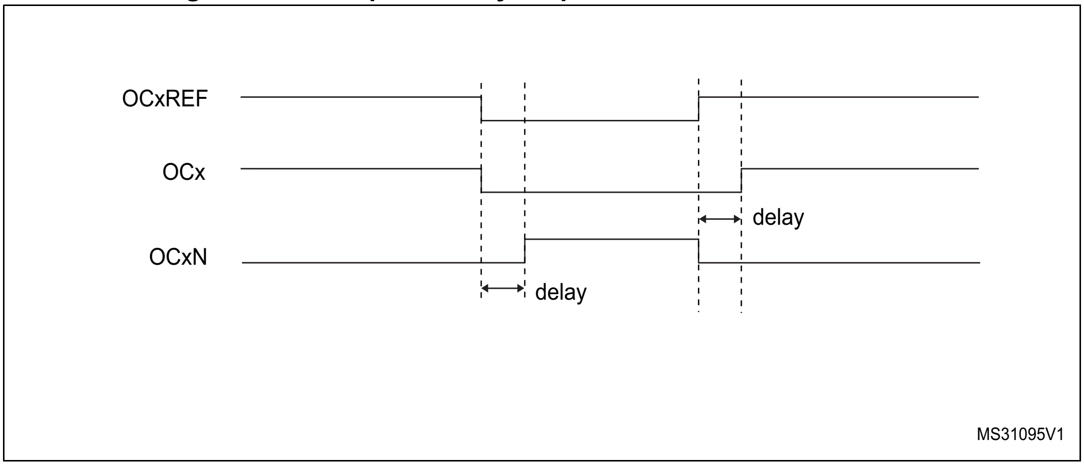


Figure 471. Dead-time waveforms with delay greater than the negative pulse.

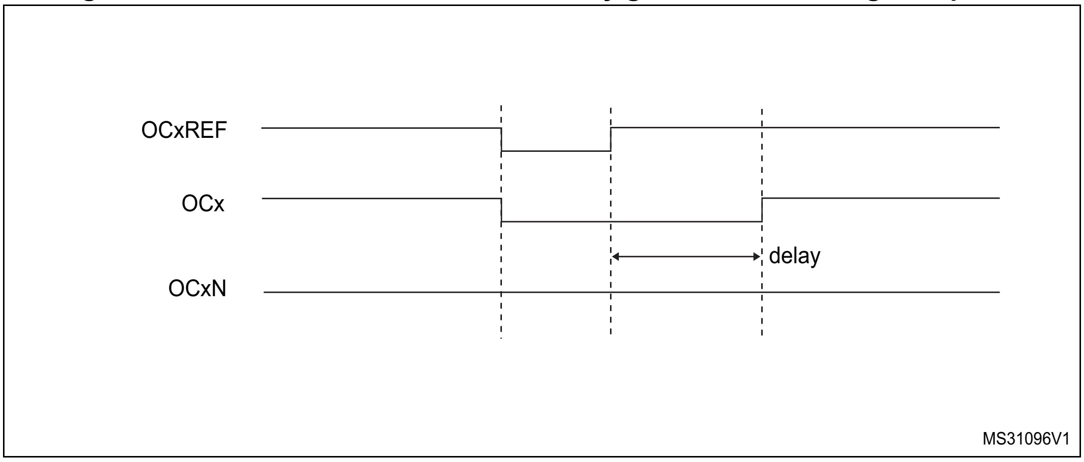
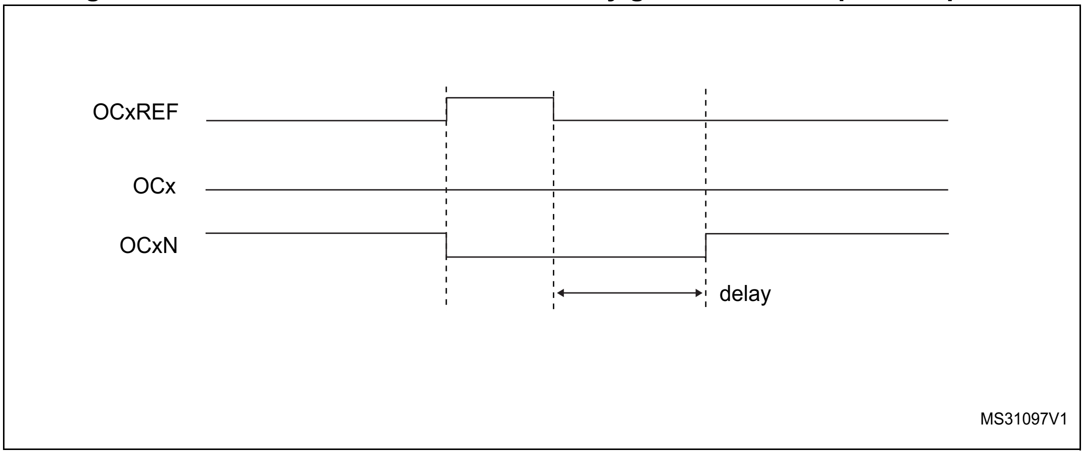


Figure 472. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 41.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 2210* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This allows a specific waveform to be sent (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.

41.4.13 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the TIM15/TIM16/TIM17 timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (clock failure, parity error,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- the MOE bit in TIMx_BDTR register allows to enable /disable the outputs by software and is reset in case of break event.
- the OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- the OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The OCx and OCxN outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 293: Output control bits for complementary OCx and OCxN channels with break feature (TIM15) on page 2186* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

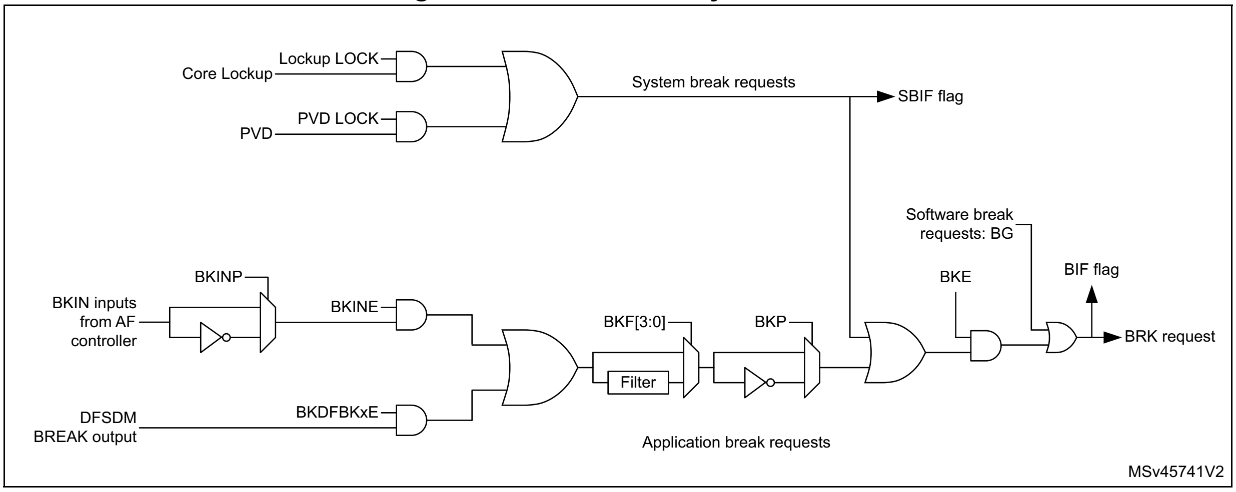
A programmable filter (BKF[3:0] bits in the TIMx_BDTR register allows to filter out spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_OR2 register.

The sources for break (BRK) channel are:

- An external source connected to one of the BKIN pin (as per selection done in the GPIO alternate function registers), with polarity selection and optional digital filtering
- An internal source:
  – the analog watchdog output of the DFSDM1 peripheral
  – A system break:
    - the Cortex®-M4 with FPU LOCKUP output
    - the PVD output

Figure 473. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE=0. If OSSI=0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value.
If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

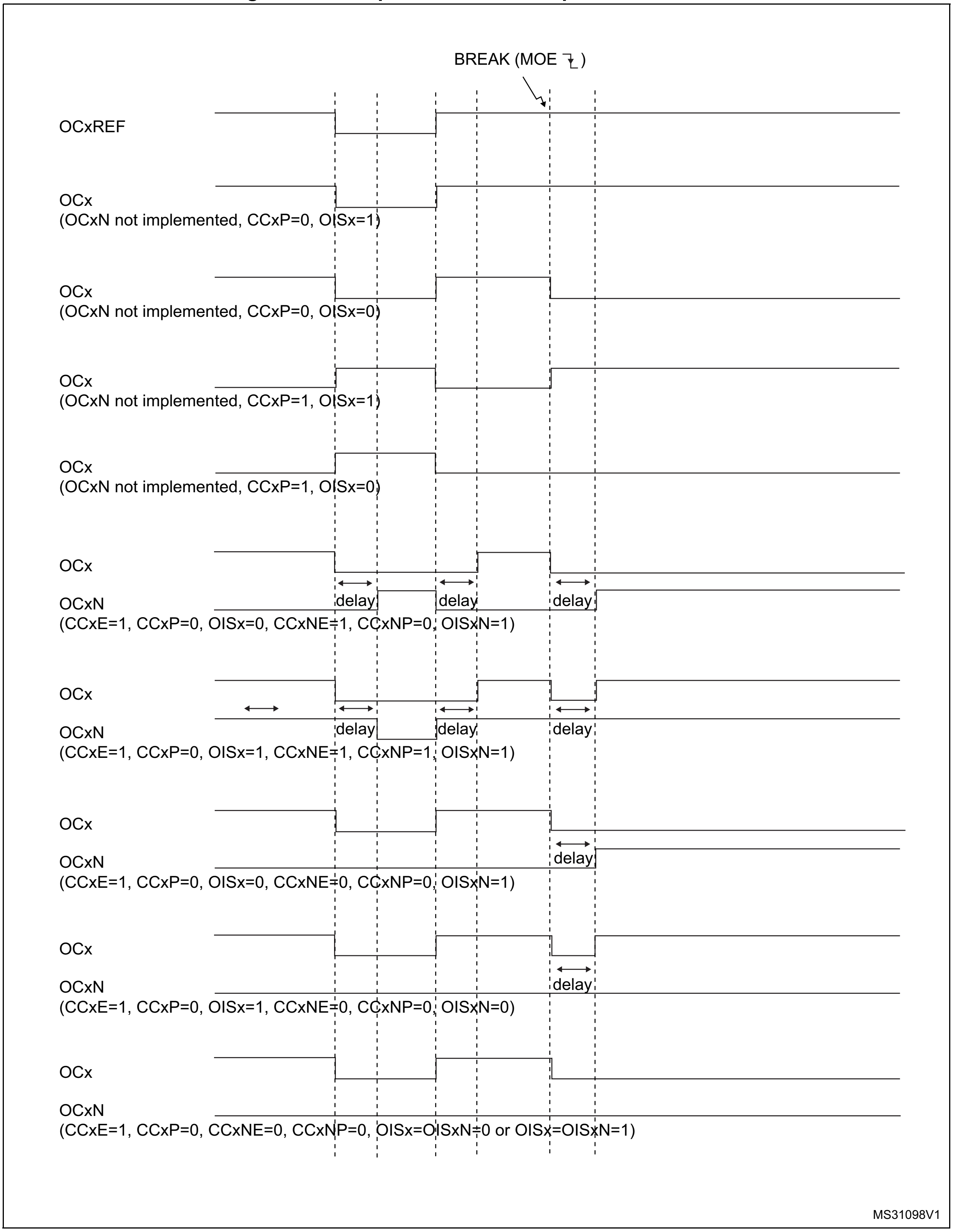
Note: The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows the configuration of several parameters to be freezed (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 41.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) on page 2210*. The LOCK bits can be written only once after an MCU reset.

The *Figure 474* shows an example of behavior of the outputs in response to a break.

Figure 474. Output behavior in response to a break



41.4.14 Bidirectional break inputs

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 475*.

They allow the following:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin
- Internal break sources and multiple external open drain comparator outputs ORed together to trigger a unique break event, when multiple internal and external break sources must be merged

The break input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (e.g. CSS), from on-chip peripherals or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (BG) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the break I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be re-started as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 291*)

Table 291. Break protection disarming conditions

MOE	BKDIR	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and re-arming break circuitry

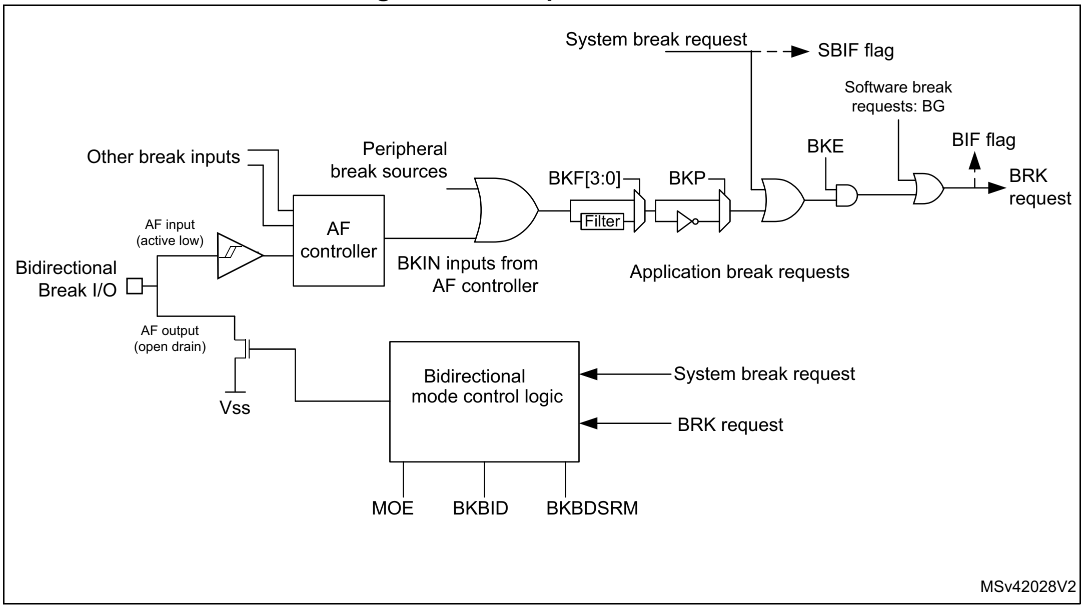
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before re-arming)
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears)

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 475. Output redirection



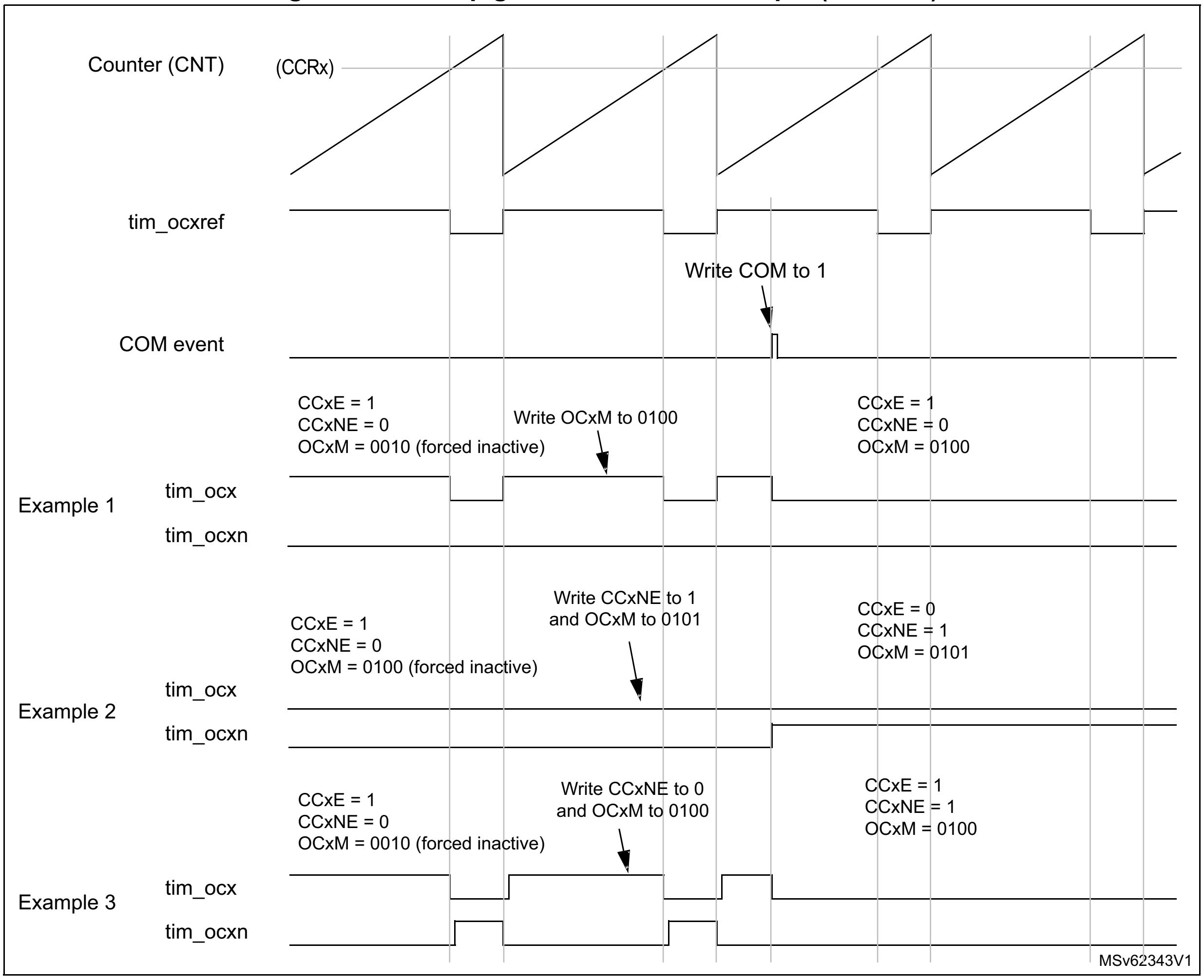
41.4.15 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 476* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 476. 6-step generation, COM example (OSSR=1)



41.4.16 One-pulse mode

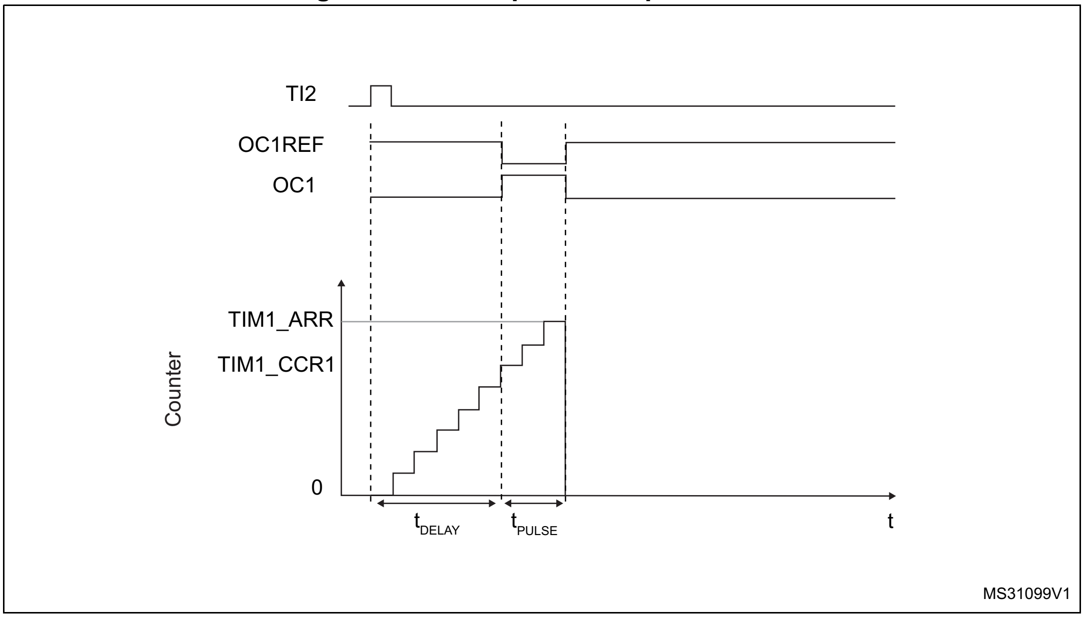
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 477. Example of one pulse mode



For example one may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

1. Select the proper TI2[x] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map TI2FP2 to TI2 by writing CC2S='01' in the TIMx_CCMR1 register.
3. TI2FP2 must detect a rising edge, write CC2P='0' and CC2NP='0' in the TIMx_CCER register.
4. Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='00110' in the TIMx_SMCR register.
5. TI2FP2 is used to start the counter by writing SMS to '110' in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let’s say one want to build a waveform with a transition from ‘0’ to ‘1’ when a compare match occurs and a transition from ‘1’ to ‘0’ when the counter reaches the auto-reload value. To do this PWM mode 2 must be enabled by writing OC1M=111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE=’1’ in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

Since only 1 pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

Particular case: OCx fast enable

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

41.4.17 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with Non-retriggerable one pulse mode described in *Section 41.4.16*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

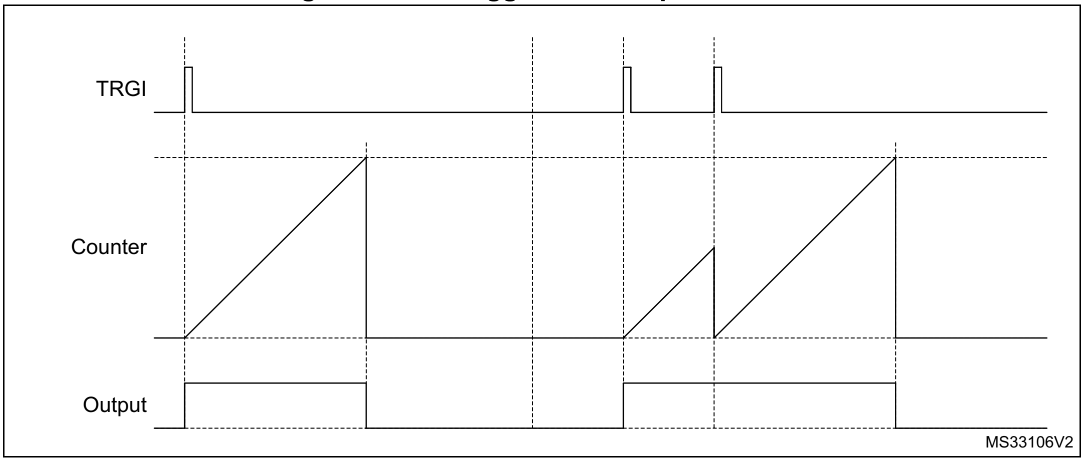
The timer must be in Slave mode, with the bits SMS[3:0] = ‘1000’ (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to ‘1000’ or ‘1001’ for Retrigerrable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the 3 least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 478. Retriggerable one pulse mode



41.4.18 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag, to be atomically read. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

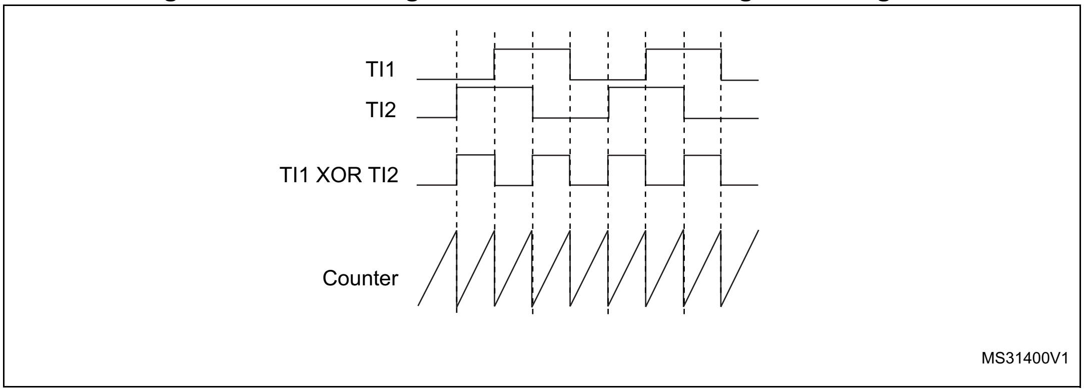
There is no latency between the assertions of the UIF and UIFCPY flags.

41.4.19 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins TIMx_CH1 and TIMx_CH2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 479*.

Figure 479. Measuring time interval between edges on 2 signals



41.4.20 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

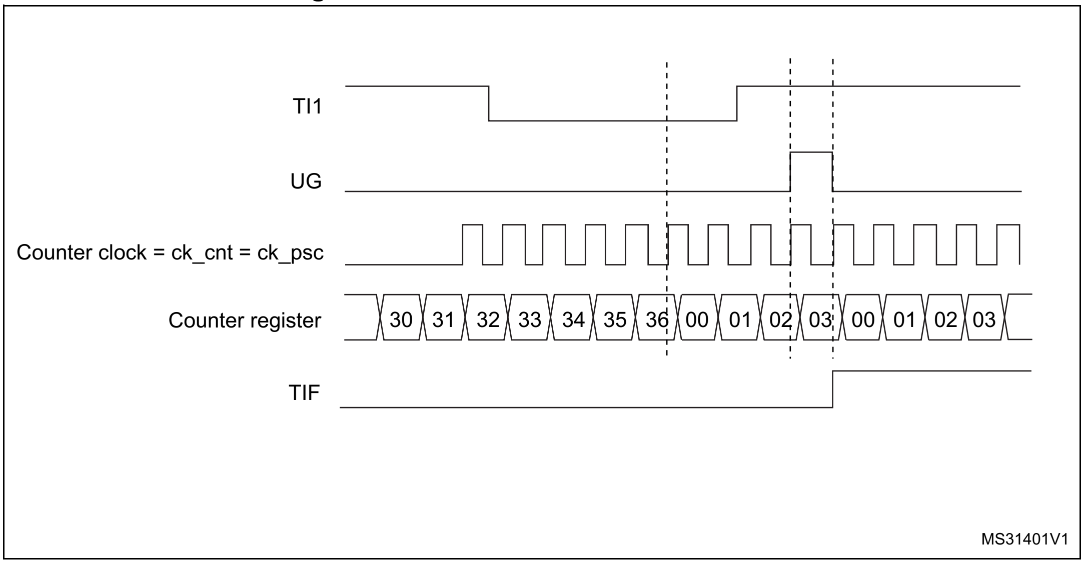
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

1. Configure the channel 1 to detect rising edges on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P='0' and CC1NP='0' in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS=100 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Start the counter by writing CEN=1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 480. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

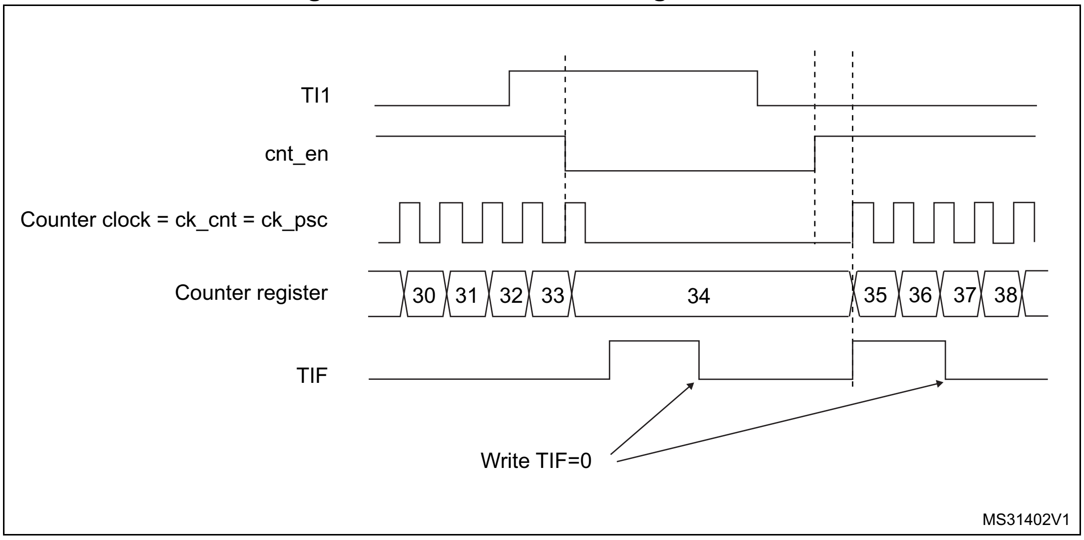
In the following example, the upcounter counts only when TI1 input is low:

1. Configure the channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S=01 in TIMx_CCMR1 register. Write CC1P=1 and CC1NP = ‘0’ in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS=101 in TIMx_SMCR register. Select TI1 as the input source by writing TS=00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN=1 in the TIMx_CR1 register (in gated mode, the counter doesn’t start if CEN=0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 481. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

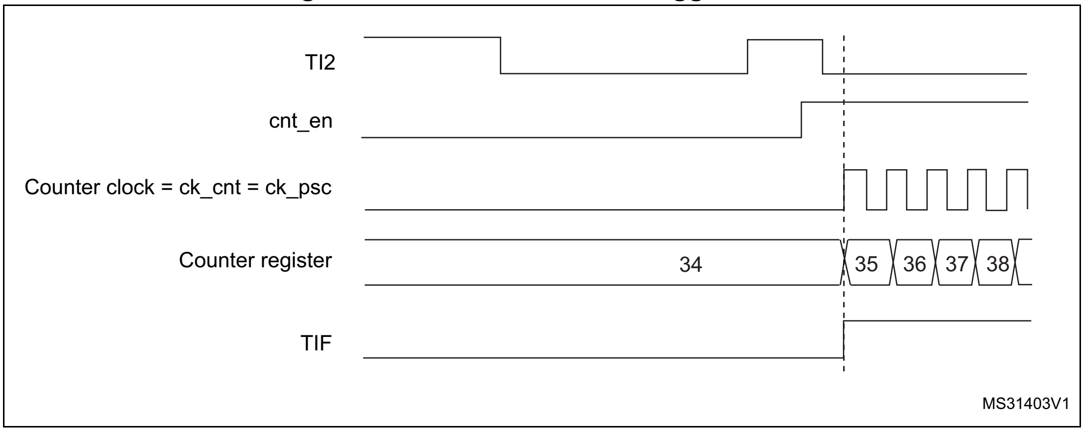
In the following example, the upcounter starts in response to a rising edge on TI2 input:

1. Configure the channel 2 to detect rising edges on TI2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S=01 in TIMx_CCMR1 register. Write CC2P='1' and CC2NP='0' in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS=110 in the TIMx_SMCR register. Select TI2 as the input source by writing TS=00110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on TI2 input.

Figure 482. Control circuit in trigger mode



41.4.21 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

41.4.22 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to re-program several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

For example, the timer DMA burst feature could be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   – Number of data to transfer = 3 (See note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer should be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5 and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3 and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

41.4.23 Timer synchronization (TIM15)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 39.3.19: Timer synchronization* for details.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

41.4.24 Using timer output as trigger for other timers (TIM16/TIM17)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any TIMx_SMCR register on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least 2 clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is 4 times slower than the source timer, the OC1 pulse width must be 8 clock cycles.

41.4.25 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on TIMx bit in DBGMCU module. For more details, refer to *Section 64.10.9: Microprocessor debug unit (DBGMCU)*.

For safety purposes, when the counter is stopped (TIMx = 1 in DBGMCU_D2APB2FZ1), the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

41.5 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

41.5.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (TIx)
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 * t_{CK\_INT}$
10: $t_{DTS} = 4 * t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

41.5.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (OC2 output)

0: OC2=0 when MOE=0
1: OC2=1 when MOE=0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BDTR register).

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **TI1S**: TI1 selection

0: The TIMx_CH1 pin is connected to TI1 input

1: The TIMx_CH1, CH2 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC1REFC signal is used as trigger output (TRGO).

101: **Compare** - OC2REFC signal is used as trigger output (TRGO).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

41.5.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bit field selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (ITR0)

00001: Internal Trigger 1 (ITR1)

00010: Internal Trigger 2 (ITR2)

00011: Internal Trigger 3 (ITR3)

00100: TI1 Edge Detector (TI1F_ED)

00101: Filtered Timer Input 1 (TI1FP1)

00110: Filtered Timer Input 2 (TI2FP2)

Other: Reserved

See *Table 292: TIMx Internal trigger connection on page 2176* for more details on ITRx meaning for each Timer.

Note: These bits must be changed only when they are not used (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update of the registers.
0101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.
0111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter, generates an update of the registers and starts the counter.
Other codes: reserved.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='00100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the TRGO or the TRGO2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

Table 292. TIMx Internal trigger connection

Slave TIM	ITR0 (TS = 00000)	ITR1 (TS = 00001)	ITR2 (TS = 00010)	ITR3 (TS = 00011)
TIM15	TIM1	TIM3	TIM16 OC1	TIM17 OC1

41.5.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw				rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

41.5.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 41.5.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS=0 and UDIS=0 in the TIMx_CR1 register.

41.5.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

41.5.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on TI2
10: CC2 channel is configured as input, IC2 is mapped on TI1
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1). The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
10: CC1 channel is configured as input, IC1 is mapped on TI2
11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

41.5.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output Compare 2 mode

Bit 11 **OC2PE**: Output Compare 2 preload enable

Bit 10 **OC2FE**: Output Compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output.
01: CC2 channel is configured as input, IC2 is mapped on TI2.
10: CC2 channel is configured as input, IC2 is mapped on TI1.
11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on TRGI signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - OC1REF has the same behavior as in PWM mode 1. OC1REFC is the logical OR between OC1REF and OC2REF.

1101: Combined PWM mode 2 - OC1REF has the same behavior as in PWM mode 2. OC1REFC is the logical AND between OC1REF and OC2REF.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

On channels that have a complementary output, this bit field is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, IC1 is mapped on TI1.

10: CC1 channel is configured as input, IC1 is mapped on TI2.

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

41.5.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:
0: OC1N active high
1: OC1N active low
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.
CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin
When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 293* for details.

Table 293. Output control bits for complementary OCx and OCxN channels with break feature (TIM15)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>OCx output state</th><th>OCxN output state</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
OCx=0
OCxN=0, OCxN_EN=0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCx=0</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF XOR CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
OCxN=0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
OCx=CCxP</td><td>OCxREF + Polarity
OCxN=OCxREF XOR CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>OCxREF + Polarity
OCx=OCxREF xor CCxP, OCx_EN=1</td><td>Off-State (output enabled with inactive state)
OCxN=CCxNP, OCxN_EN=1</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: OCx=CCxP, OCxN=CCxNP
Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and GPIO control and alternate function registers.

41.5.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

41.5.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

41.5.12 TIM15 auto-reload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 41.4.1: Time-base unit on page 2136* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

41.5.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

41.5.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

41.5.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR2[15:0]**: Capture/Compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (IC2).

41.5.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode

1: Break input BRK in bidirectional mode

In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample the BRK input signal and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)
See OC/OCN enable description for more details (*Section 41.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2184*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled

1; Break inputs (BRK and CCS clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 41.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2184*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 41.5.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2184*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x t_{dtg} with t_{dtg} = t_{DTS}

DTG[7:5] = 10x => DT = (64+DTG[5:0]) x t_{dtg} with t_{dtg} = 2 x t_{DTS}

DTG[7:5] = 110 => DT = (32+DTG[4:0]) x t_{dtg} with t_{dtg} = 8 x t_{DTS}

DTG[7:5] = 111 => DT = (32+DTG[4:0]) x t_{dtg} with t_{dtg} = 16 x t_{DTS}

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

41.5.17 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

00000: 1 transfer,

00001: 2 transfers,

00010: 3 transfers,

...

10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

41.5.18 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

41.5.19 TIM15 alternate register 1 (TIM15_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	BKINP	BKDF1 BK0E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
						rw	rw								rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK0E**: BRK dfsdm1_break[0] enable

This bit enables the dfsdm1_break[0] for the timer's BRK input. dfsdm1_break[0] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[0]input disabled

1: dfsdm1_break[0]input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

41.5.20 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects TI2[0] to TI2[15] input
0000: TIM15_CH2 input
0001: TIM2_CH2 input
0010: TIM3_CH2 input
0011: TIM4_CH2 input
Others: Reserved

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM15_CH1 input
0001: TIM2_CH1 input
0010: TIM3_CH1 input
0011: TIM4_CH1 input
0100: LSE
0101: CSI
0110: MCO2
0111: hsi_cal_ck
1000: csi_cal_ck
Other: Reserved

41.5.21 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 294. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0

Table 294. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	**TIM15_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	**TIM15_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0				0	0	0	0	0			0	0	0
0x10	**TIM15_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	**TIM15_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	**TIM15_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM15_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM15_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM15_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM15_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 294. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	TIM15_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIM15_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM15_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIM15_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIM15_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIM15_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIM15_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINP	BKDF1BK0E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
	Reset value																							0	0								1
0x68	TIM15_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

41.6 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

41.6.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (TIx),
00: $t_{DTS} = t_{CK\_INT}$
01: $t_{DTS} = 2 * t_{CK\_INT}$
10: $t_{DTS} = 4 * t_{CK\_INT}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

41.6.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (OC1N output)

0: OC1N=0 after a dead-time when MOE=0
1: OC1N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output Idle state 1 (OC1 output)

0: OC1=0 (after a dead-time if OC1N is implemented) when MOE=0
1: OC1=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

Note: This bit acts only on channels that have a complementary output.

41.6.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable

0: CC1 DMA request disabled

1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled

1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

0: Break interrupt disabled

1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable

0: COM interrupt disabled

1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

41.6.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to ‘0’.
0: No overcapture has been detected
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
- At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS=0 and UDIS=0 in the TIMx_CR1 register.

41.6.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action.
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

41.6.6 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC1 input (IC1).
The prescaler is reset as soon as CC1E='0' (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 Selection
This bit-field defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, IC1 is mapped on TI1
Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

41.6.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal OC1REF from which OC1 and OC1N are derived. OC1REF is active high whereas OC1 and OC1N active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. OC1REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. OC1REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - OC1REF toggles when TIMx_CNT=TIMx_CCR1.

0100: Force inactive level - OC1REF is forced low.

0101: Force active level - OC1REF is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

All other values: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

The OC1M[3] bit is not contiguous, located in bit 16.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S='00' (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in TIMx_CCER).

41.6.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: OC1N active high

1: OC1N active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of TI1FP1 and TI2FP1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S="00" (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - OC1N is not active. OC1N level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - OC1N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP=0, CC1P=0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP=0, CC1P=1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP=1, CC1P=1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP=1, CC1P=0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 295* for details.

Table 295. Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)

Control bits					Output states[(1)]	
MOE bit	**OSSI bit**	**OSSR bit**	**CCxE bit**	**CCxNE bit**	**OCx output state**	**OCxN output state**
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) OCx=0 OCxN=0, OCxN_EN=0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) OCx=0	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		0	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) OCxN=0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP	OCxREF + Polarity OCxN=OCxREF XOR CCxNP
		1	1	0	OCxREF + Polarity OCx=OCxREF XOR CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCxN=CCxNP Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and GPIO control and alternate function registers.

41.6.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

41.6.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

41.6.11 TIMx auto-reload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 41.4.1: Time-base unit on page 2136* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

41.6.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode.

41.6.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **CCR1[15:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

41.6.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input BRK in input mode
1: Break input BRK in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input BRK is armed
1: Break input BRK is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bit-field defines the frequency used to sample BRK input and the length of the digital filter applied to BRK. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, BRK acts asynchronously
0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2
0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4
0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8
0100: $f_{SAMPLING}=f_{DTS}/2$, N=6
0101: $f_{SAMPLING}=f_{DTS}/2$, N=8
0110: $f_{SAMPLING}=f_{DTS}/4$, N=6
0111: $f_{SAMPLING}=f_{DTS}/4$, N=8
1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.
1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register)

See OC/OCN enable description for more details (*Section 41.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 2205*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low
1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CCS clock failure event) disabled
1; Break inputs (BRK and CCS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 41.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 2205*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section 41.6.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 2205*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0)

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0] x t_{dtg} with t_{dtg} = t_{DTS}

DTG[7:5] = 10x => DT = (64 + DTG[5:0]) x t_{dtg} with t_{dtg} = 2 x t_{DTS}

DTG[7:5] = 110 => DT = (32 + DTG[4:0]) x t_{dtg} with t_{dtg} = 8 x t_{DTS}

DTG[7:5] = 111 => DT = (32 + DTG[4:0]) x t_{dtg} with t_{dtg} = 16 x t_{DTS}

Example if t_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

41.6.15 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit field defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bit field defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

41.6.16 TIMx DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **DMAB[15:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

41.6.17 TIM16 alternate function register 1 (TIM16_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	BKINP	BKDF1 BK1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
						rw	rw								rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BKINP**: BRK BKIN input polarity

This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: BKIN input is active low

1: BKIN input is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK1E**: BRK dfsdm1_break[1] enable

This bit enables the dfsdm1_break[1] for the timer's BRK input. dfsdm1_break[1] output is 'ORed' with the other BRK sources.

0: dfsdm1_break[1] input disabled

1: dfsdm1_break[1] input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable

This bit enables the BKIN alternate function input for the timer's BRK input. BKIN input is 'ORed' with the other BRK sources.

0: BKIN input disabled

1: BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

41.6.18 TIM16 input selection register (TIM16_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input
0000: TIM16_CH1 input
0001: LSI
0010: LSE
0011: WKUP_IT
Other: Reserved

41.6.19 TIM17 alternate function register 1 (TIM17_AF1)

Address offset: 0x60

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	BKINP	BKDF1 BK2E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
						rw	rw								rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **BKINP**: BRK BKIN input polarity
This bit selects the BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: BKIN input is active low
1: BKIN input is active high
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKDF1BK2E**: BRK dfsdm1_break[2] enable
This bit enables the dfsdm1_break[2] for the timer’s BRK input. dfsdm1_break[2] output is ‘ORed’ with the other BRK sources.
0: dfsdm1_break[2] input disabled
1: dfsdm1_break[2] input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BKINE**: BRK BKIN input enable
This bit enables the BKIN alternate function input for the timer’s BRK input. BKIN input is ‘ORed’ with the other BRK sources.
0: BKIN input disabled
1: BKIN input enabled
Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

41.6.20 TIM17 input selection register (TIM17_TISEL)

Address offset: 0x68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects TI1[0] to TI1[15] input

0000: TIM17_CH1 input

0001: SPDIFRX_FRAME_SYNC is connected to TIM17_CH1 to measure the clock drift of the received SPDIF frames

0010: HSE_RTC

0011: MCO1

Other: Reserved

41.6.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 296. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	**TIMx_CCMR1** Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0										0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x20	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 296. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TIMx_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DTG[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x4C	TIMx_DMAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMAB[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	TIM16_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINP	BKDF1BK1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
	Reset value																							0	0								1
0x60	TIM17_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINP	BKDF1BK2E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKINE
	Reset value																							0	0								1
0x68	TIM16_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x68	TIM17_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0

Refer to Section 2.5 on page 153 for the register boundary addresses.

42 Basic timers (TIM6/TIM7)

42.1 TIM6/TIM7 introduction

The basic timers TIM6 and TIM7 consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used as generic timers for time base generation but they are also specifically used to drive the digital-to-analog converter (DAC). In fact, the timers are internally connected to the DAC and are able to drive it through their trigger outputs.

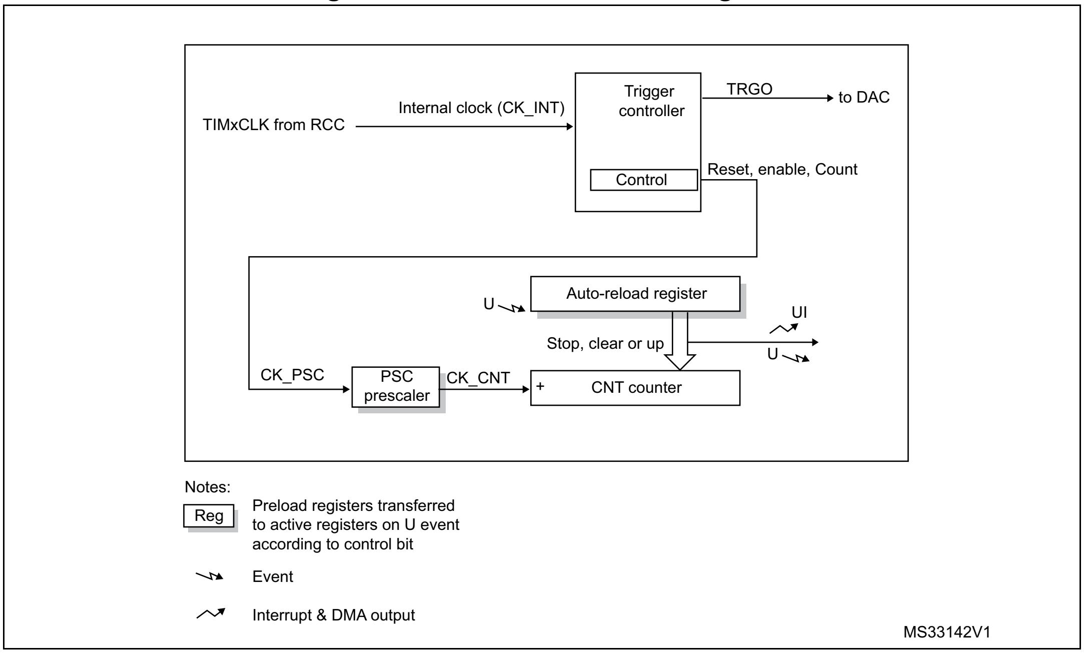
The timers are completely independent, and do not share any resources.

42.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:

- 16-bit auto-reload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Synchronization circuit to trigger the DAC
- Interrupt/DMA generation on the update event: counter overflow

Figure 483. Basic timer block diagram



42.3 TIM6/TIM7 functional description

42.3.1 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related auto-reload register. The counter clock can be divided by a prescaler.

The counter, the auto-reload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter Register (TIMx_CNT)
- Prescaler Register (TIMx_PSC)
- Auto-Reload Register (TIMx_ARR)

The auto-reload register is preloaded. The preload register is accessed each time an attempt is made to write or read the auto-reload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal CNT_EN is set 1 clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 484 and *Figure 485* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 484. Counter timing diagram with prescaler division change from 1 to 2

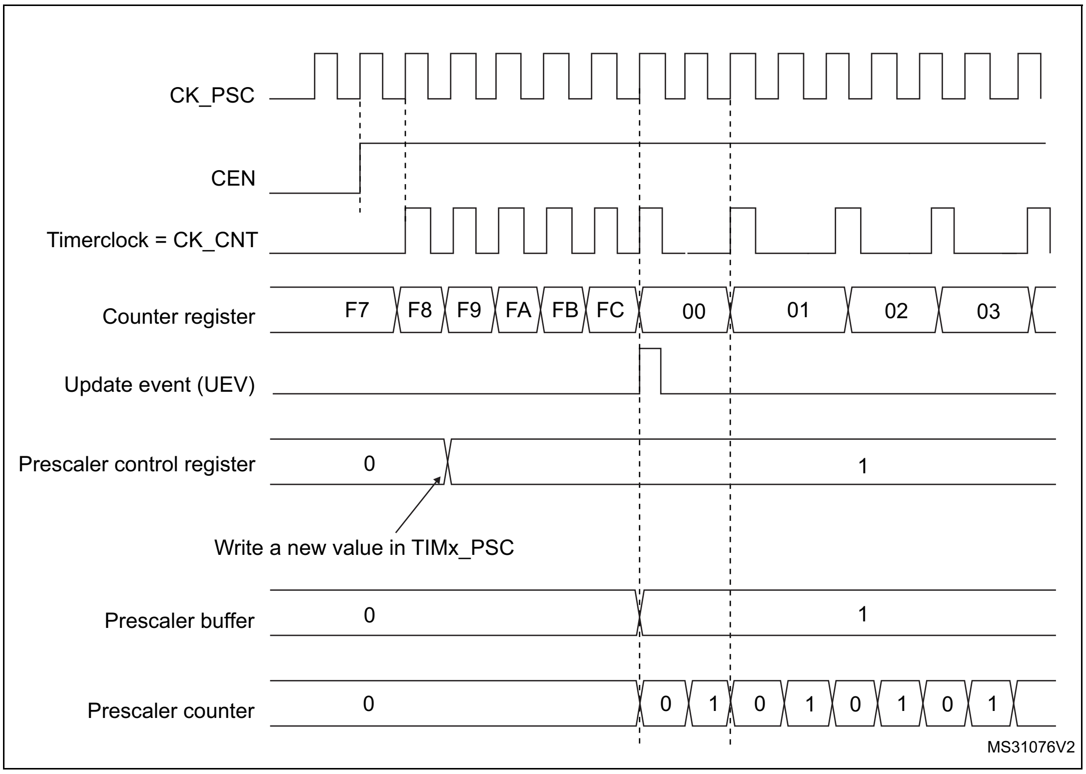
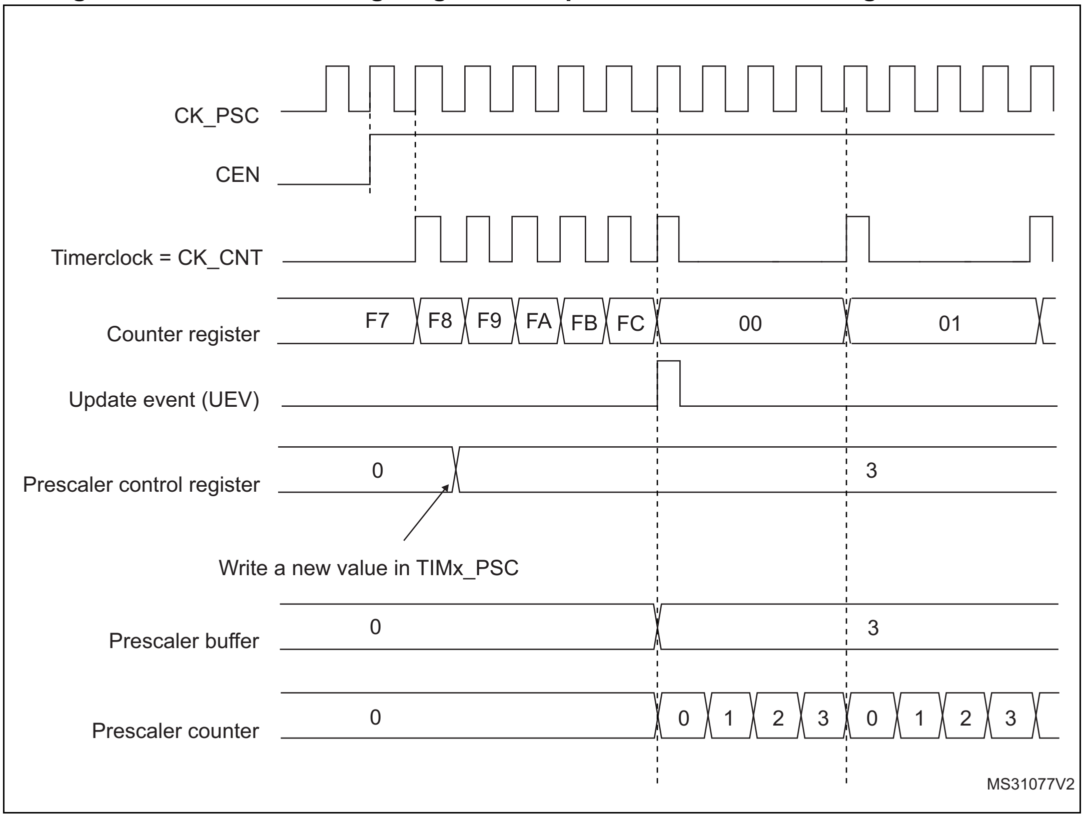


Figure 485. Counter timing diagram with prescaler division change from 1 to 4



42.3.2 Counting mode

The counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been cleared, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register)
- The auto-reload shadow register is updated with the preload value (TIMx_ARR)

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 486. Counter timing diagram, internal clock divided by 1

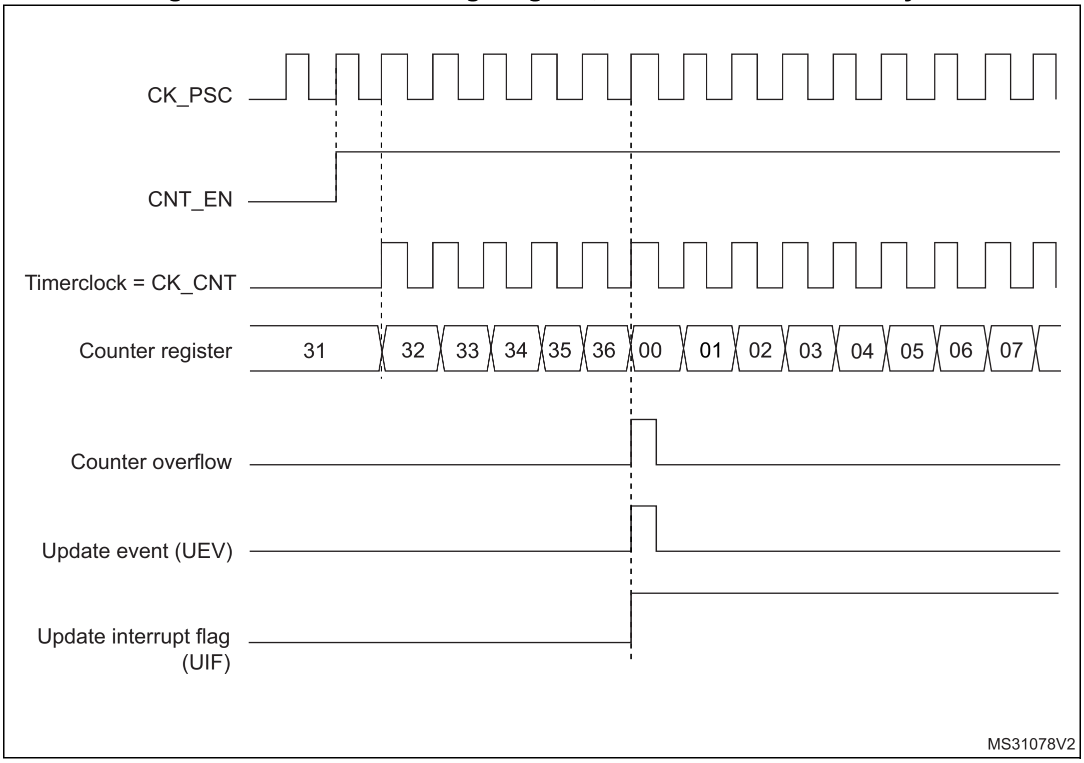


Figure 487. Counter timing diagram, internal clock divided by 2

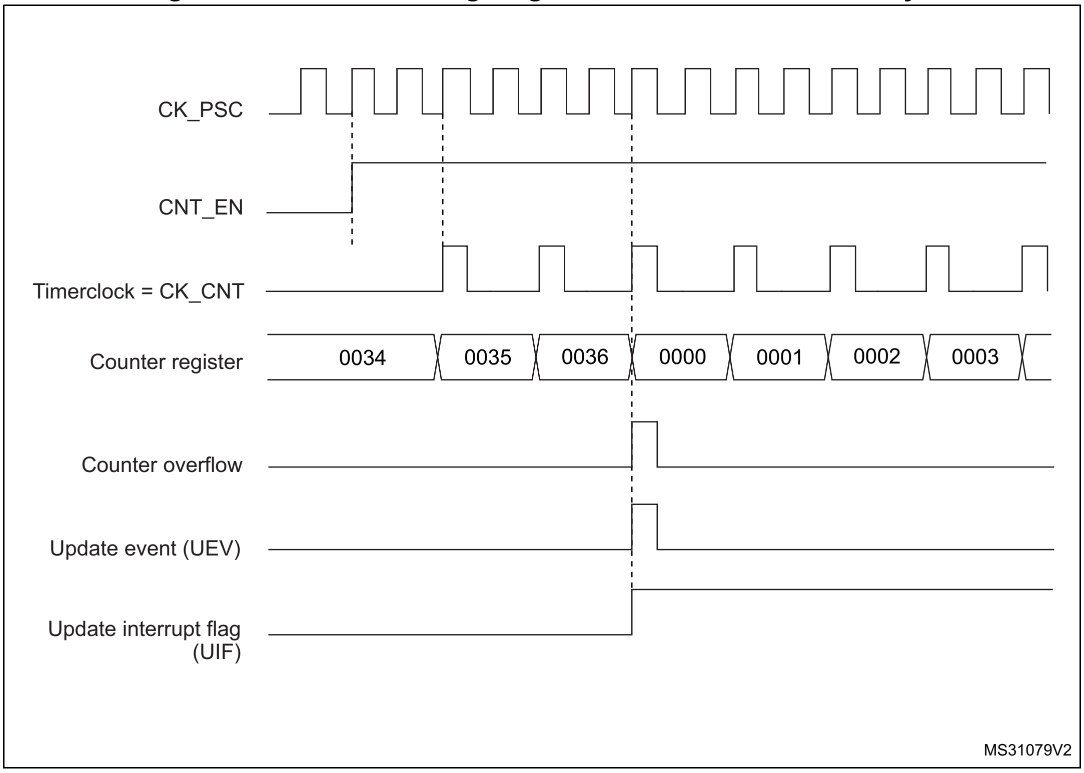


Figure 488. Counter timing diagram, internal clock divided by 4

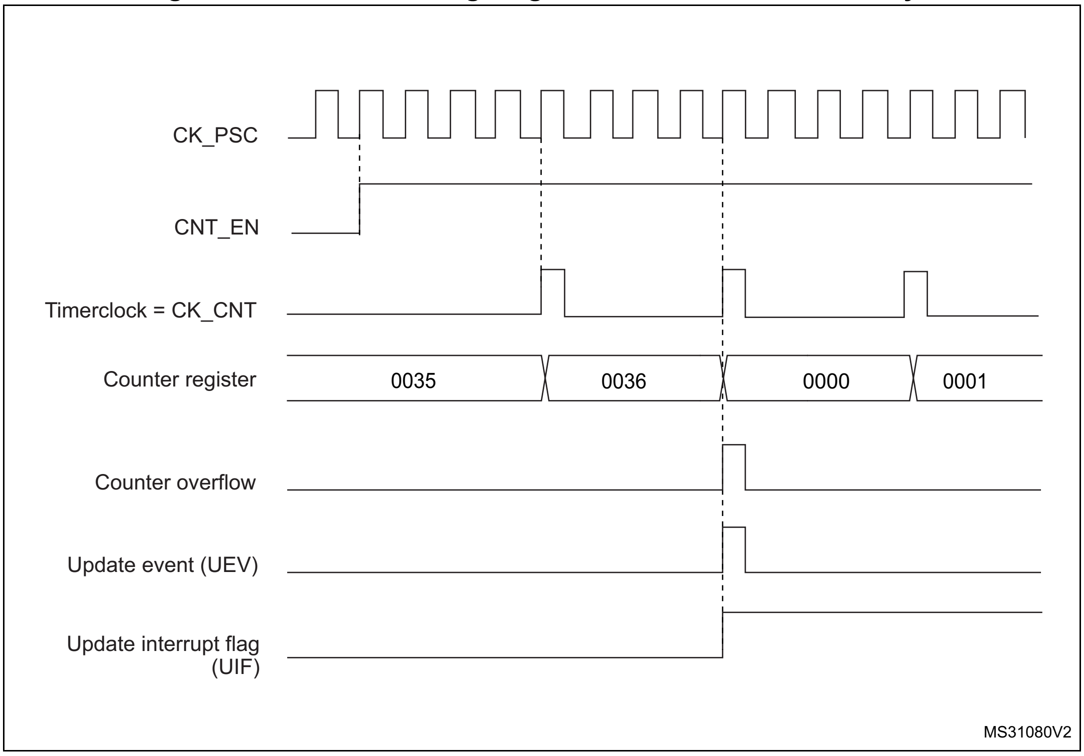


Figure 489. Counter timing diagram, internal clock divided by N

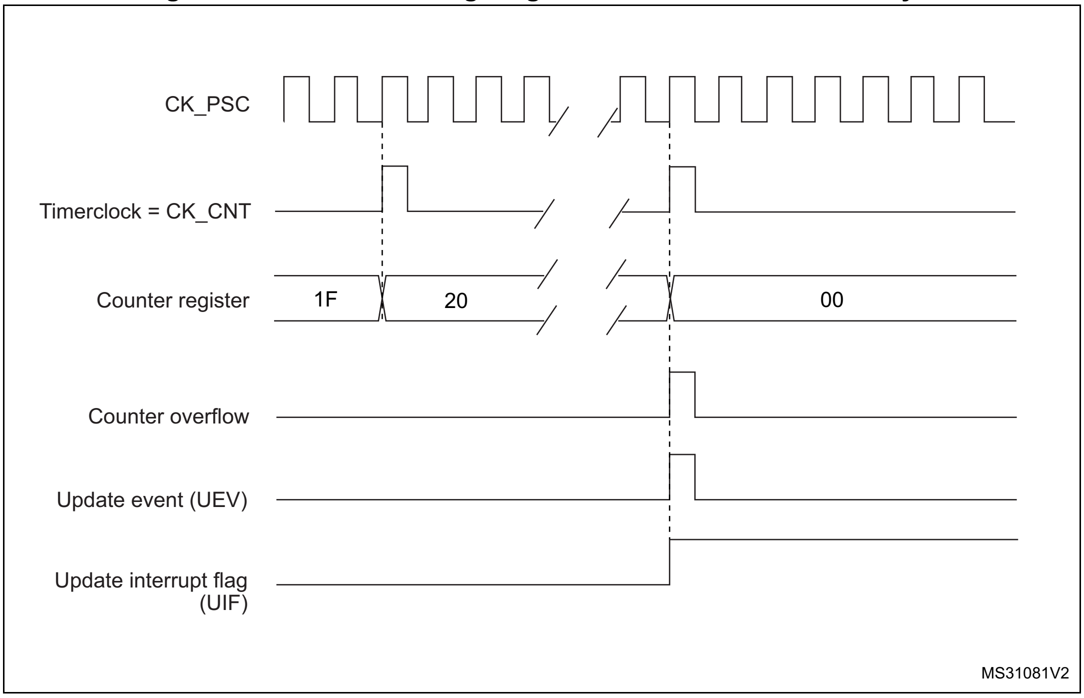


Figure 490. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

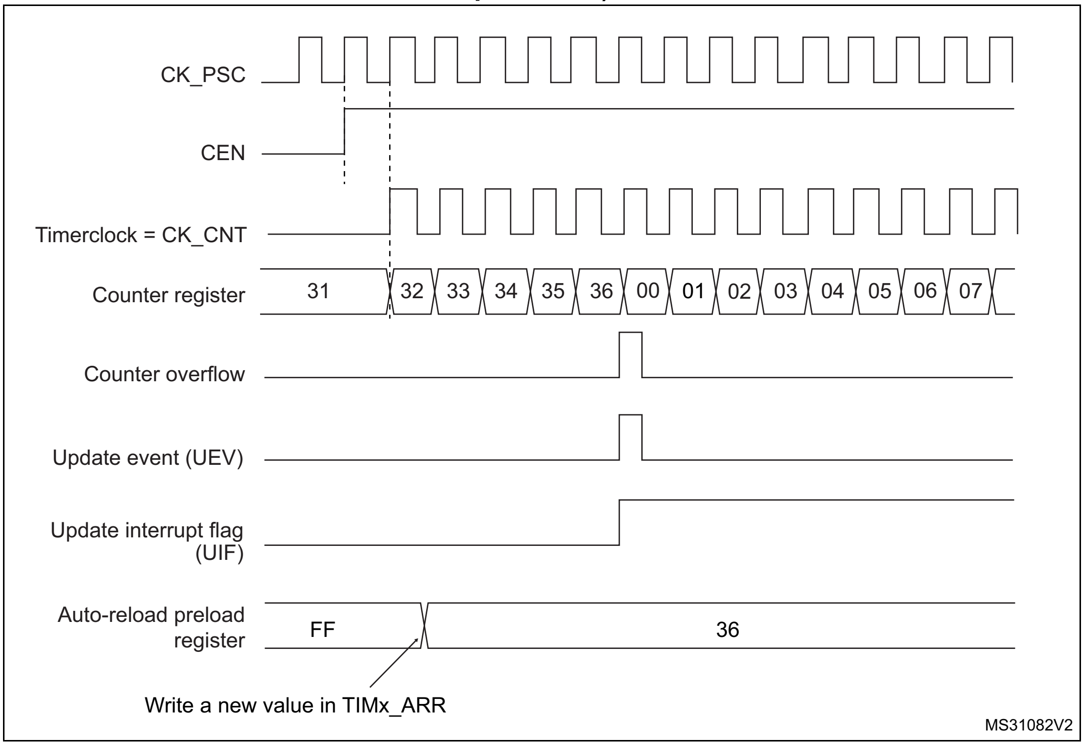
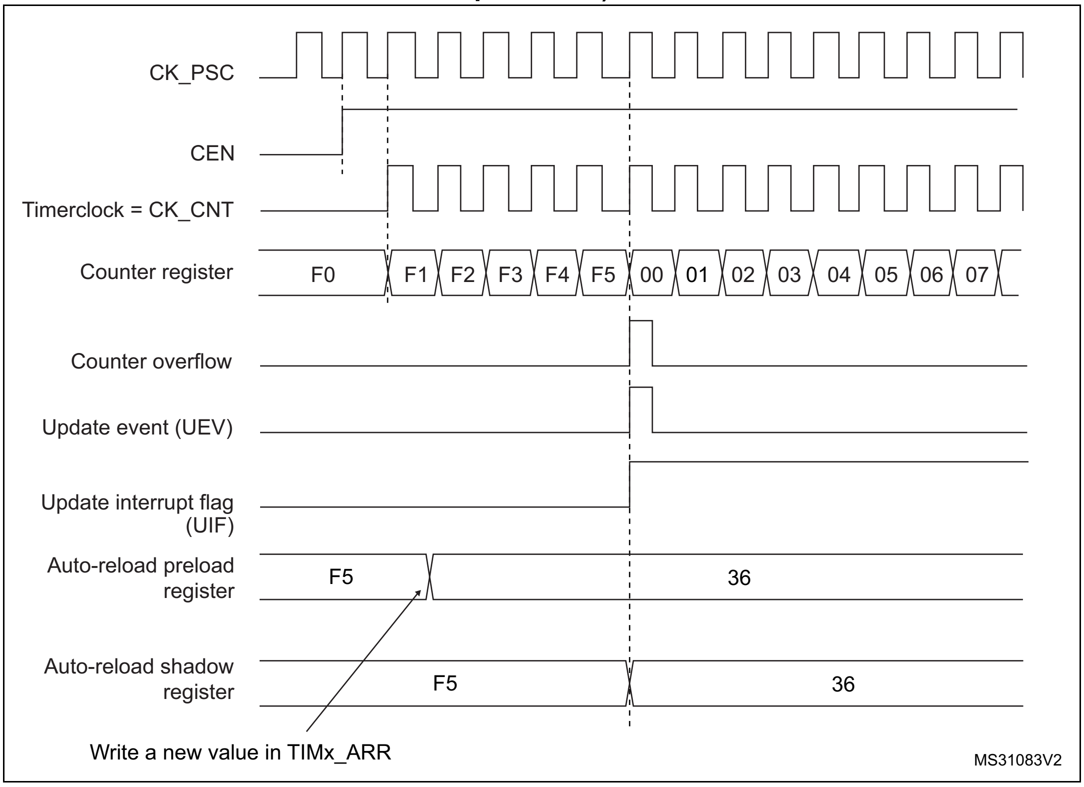


Figure 491. Counter timing diagram, update event when ARPE=1 (TIMx_ARR preloaded)



42.3.3 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the Update Interrupt Flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (Update Interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

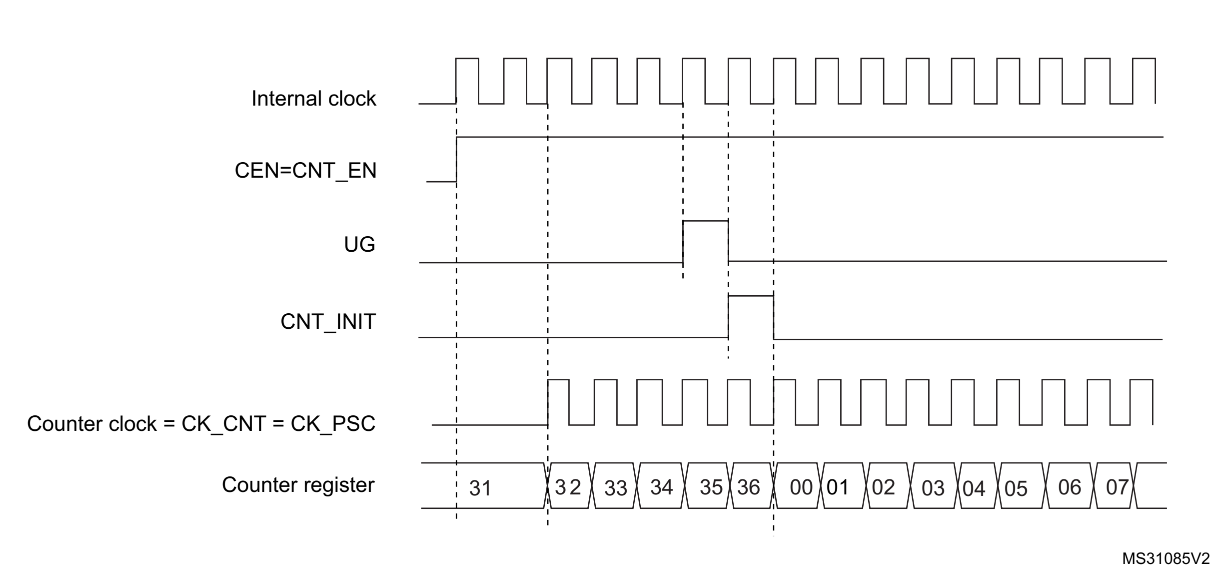
42.3.4 Clock source

The counter clock is provided by the Internal clock (CK_INT) source.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 492 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 492. Control circuit in normal mode, internal clock divided by 1



42.3.5 Debug mode

When the system enters debug mode (processor core halted), the TIMx counter either continues to work normally or stops, depending on TIMx bit in DBGMCU module. For more details, refer to *Section 64.10.9: Microprocessor debug unit (DBGMCU)*.

42.4 TIM6/TIM7 registers

Refer to *Section 1.2 on page 115* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UIFRE MAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
				rw				rw				rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled
1: Counter enabled

Note: Gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

42.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on TRGO is delayed compared to the actual reset.

001: **Enable** - the Counter enable signal, CNT_EN, is used as a trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic OR between CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on TRGO, except if the master/slave mode is selected (see the MSM bit description in the TIMx_SMCR register).

010: **Update** - The update event is selected as a trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

42.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.

1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.

1: Update interrupt enabled.

42.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value and if UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

42.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

42.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

42.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency CK_CNT is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

42.4.8 TIMx auto-reload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[15:0]**: Prescaler value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 42.3.1: Time-base unit on page 2220* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

42.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 297. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0				0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

43 Low-power timer (LPTIM)

43.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

43.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no embedded oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode

43.3 LPTIM implementation

Table 298 describes LPTIM implementation on STM32MP153 devices. The full set of features is implemented in LPTIM1 and LPTIM2. LPTIM3, LPTIM4 and LPTIM5 support a smaller set of features.

Table 298. STM32MP153 LPTIM features

LPTIM features	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
Encoder mode	X	X	-	-	-
External input clock	X	X	X	-	-
Wake-up from Stop + LP-Stop or LPLV-Stop (kernel clock from LSE or LSI only)	X	X	X	X	X

43.4 LPTIM functional description

43.4.1 LPTIM block diagram

Figure 493. Low-power timer block diagram (LPTIM1 and LPTIM2)

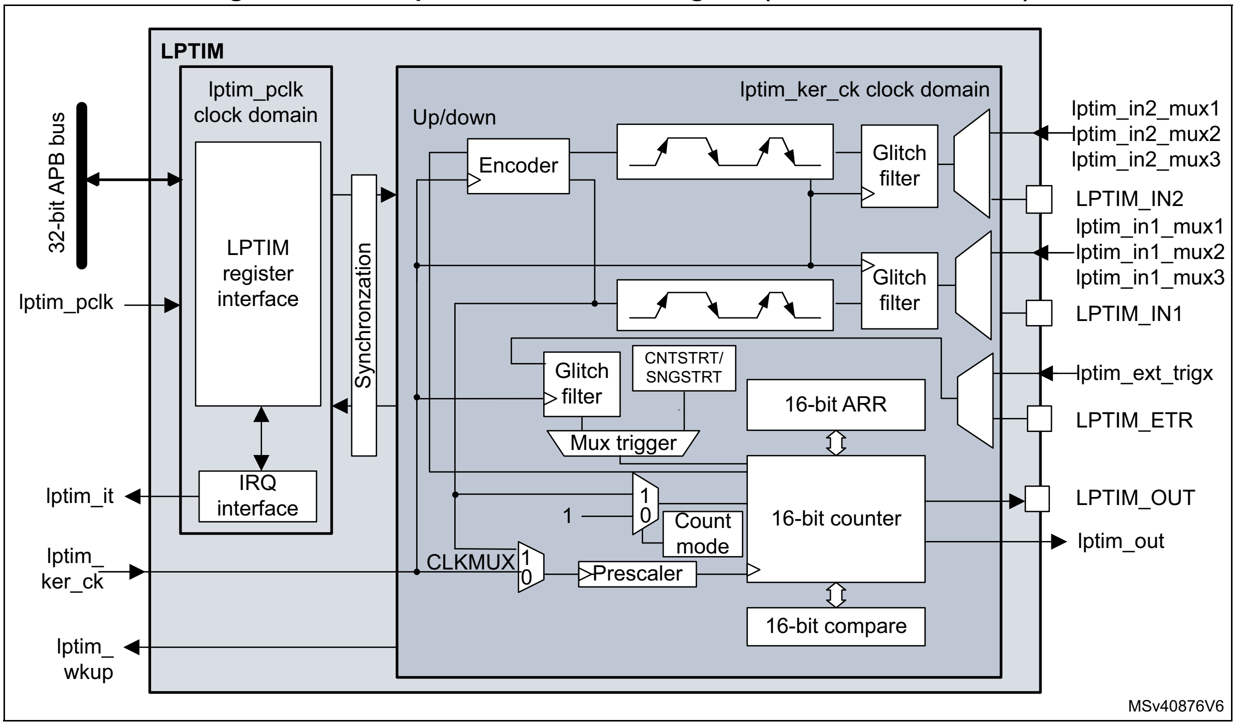


Figure 494. Low-power timer block diagram (LPTIM3)

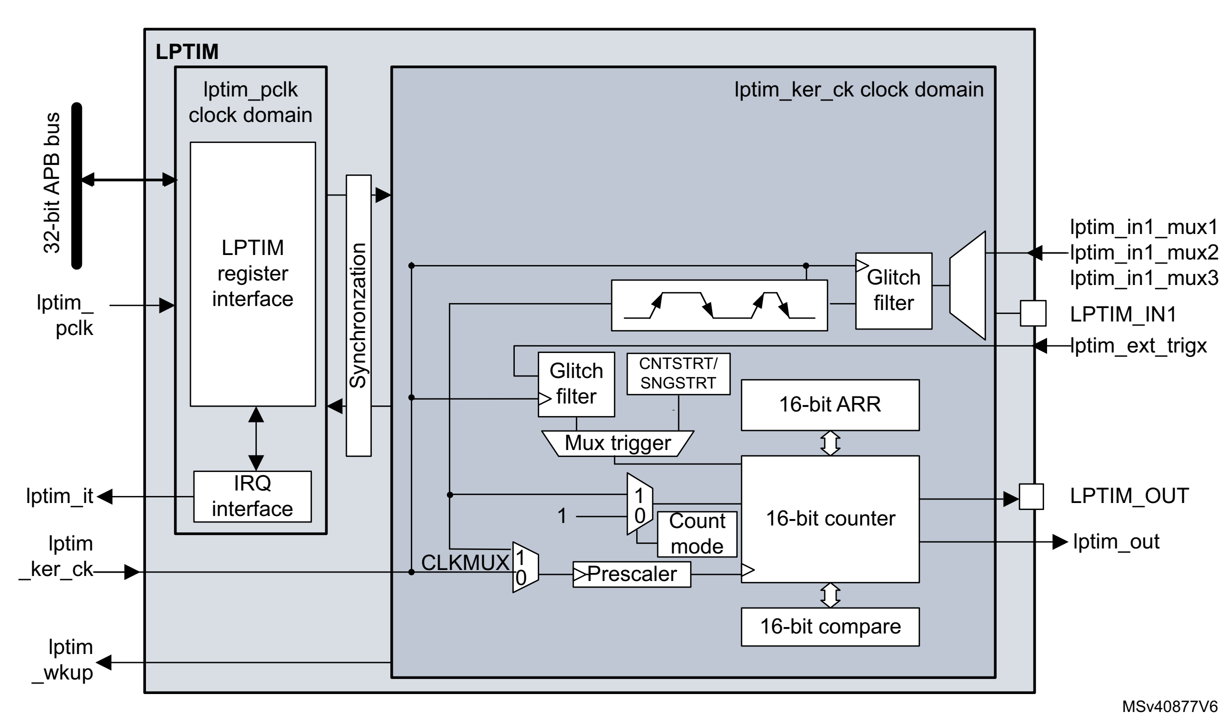
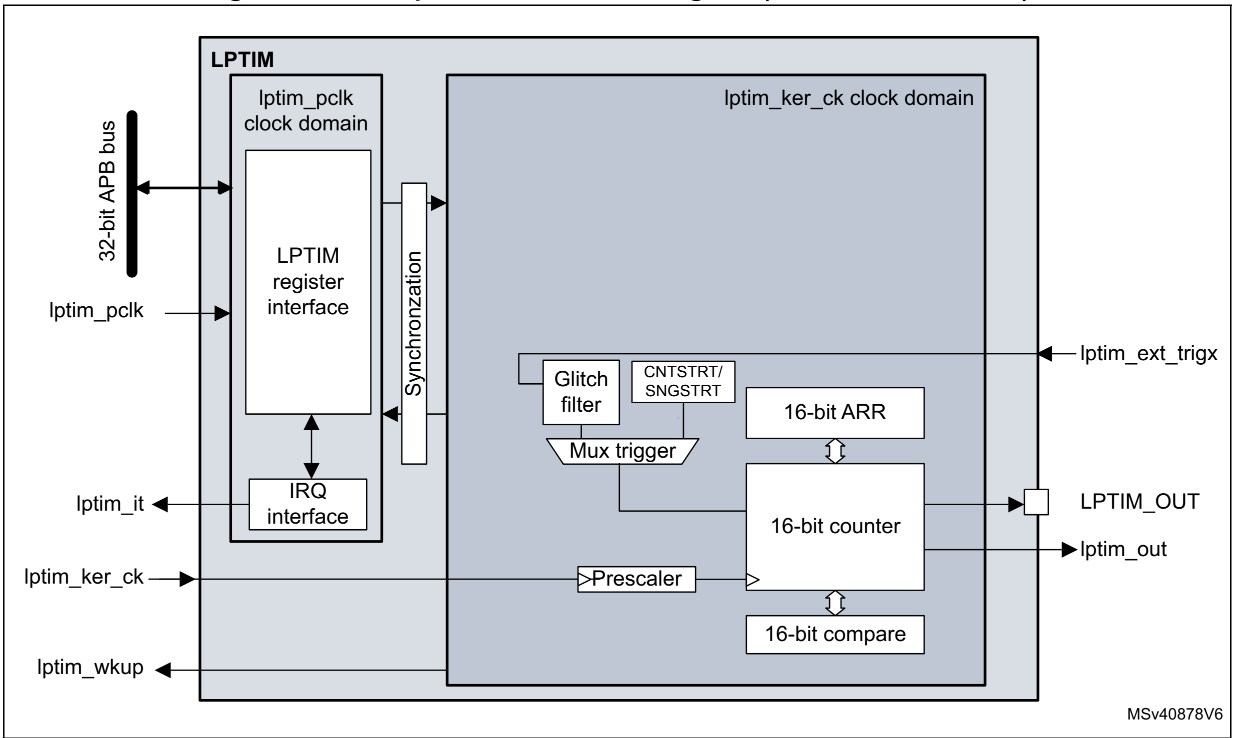


Figure 495. Low-power timer block diagram (LPTIM4 and LPTIM5)



43.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 299. LPTIM input/output pins

Names	Signal type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

Table 300. LPTIM internal signals

Names	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

43.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter:

Table 301. LPTIM1 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM1_ETR alternate function
lptim_ext_trig1	RTC_ALARMA
lptim_ext_trig2	RTC_ALARMB
lptim_ext_trig3	RTC_TAMP1_OUT
lptim_ext_trig4	RTC_TAMP2_OUT
lptim_ext_trig5	RTC_TAMP3_OUT

Table 301. LPTIM1 external trigger connection (continued)

TRIGSEL	External trigger
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 302. LPTIM2 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	GPIO pin as LPTIM2_ETR alternate function
lptim_ext_trig1	RTC_ALARMA
lptim_ext_trig2	RTC_ALARMB
lptim_ext_trig3	RTC_TAMP1_OUT
lptim_ext_trig4	RTC_TAMP2_OUT
lptim_ext_trig5	RTC_TAMP3_OUT
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 303. LPTIM3 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM2_OUT
lptim_ext_trig1	Not connected
lptim_ext_trig2	LPTIM4_OUT
lptim_ext_trig3	LPTIM5_OUT
lptim_ext_trig4	SAI1_FS_A
lptim_ext_trig5	SAI1_FS_B
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 304. LPTIM4 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM2_OUT
lptim_ext_trig1	LPTIM3_OUT
lptim_ext_trig2	Not connected
lptim_ext_trig3	LPTIM5_OUT
lptim_ext_trig4	SAI2_FS_A
lptim_ext_trig5	SAI2_FS_B

Table 304. LPTIM4 external trigger connection (continued)

TRIGSEL	External trigger
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 305. LPTIM5 external trigger connection

TRIGSEL	External trigger
lptim_ext_trig0	LPTIM2_OUT
lptim_ext_trig1	LPTIM3_OUT
lptim_ext_trig2	LPTIM4_OUT
lptim_ext_trig3	SAI4_FS_A
lptim_ext_trig4	SAI4_FS_B
lptim_ext_trig5	Not connected
lptim_ext_trig6	Not connected
lptim_ext_trig7	Not connected

Table 306. LPTIM1 input 1 connection

lptim_in1_mux	LPTIM1 input 1 connected to
lptim_in1_mux0	GPIO pin as LPTIM1_IN1 alternate function
lptim_in1_mux1	Not connected
lptim_in1_mux2	Not connected
lptim_in1_mux3	Not connected

Table 307. LPTIM1 input 2 connection

lptim_in2_mux	LPTIM1 input 2 connected to
lptim_int2_mux0	GPIO pin as LPTIM1_IN2 alternate function
lptim_in2_mux1	Not connected
lptim_in2_mux2	Not connected
lptim_in2_mux3	Not connected

Table 308. LPTIM2 input 1 connection

lptim_in1_mux	LPTIM2 input 1 connected to
lptim_in1_mux0	GPIO pin as LPTIM2_IN1 alternate function
lptim_in1_mux1	Not connected
lptim_in1_mux2	Not connected
lptim_in1_mux3	Not connected

Table 309. LPTIM2 input 2 connection

lptim_in2_mux	LPTIM2 input 2 connected to
lptim_int2_mux0	GPIO pin as LPTIM2_IN2 alternate function
lptim_in2_mux1	Not connected
lptim_in2_mux2	Not connected
lptim_in2_mux3	Not connected

Table 310. LPTIM3 input 1 connection

lptim_in1_mux	LPTIM3 Input 1 connected to
lptim_in1_mux0	Not connected
lptim_in1_mux1	SAI4_FS_A
lptim_in1_mux2	SAI4_FS_B
lptim_in1_mux3	Not connected

43.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM will use an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal should also be provided (first configuration). In this case, the internal clock signal frequency should be at least four times higher than the external clock signal frequency.

43.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source should first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

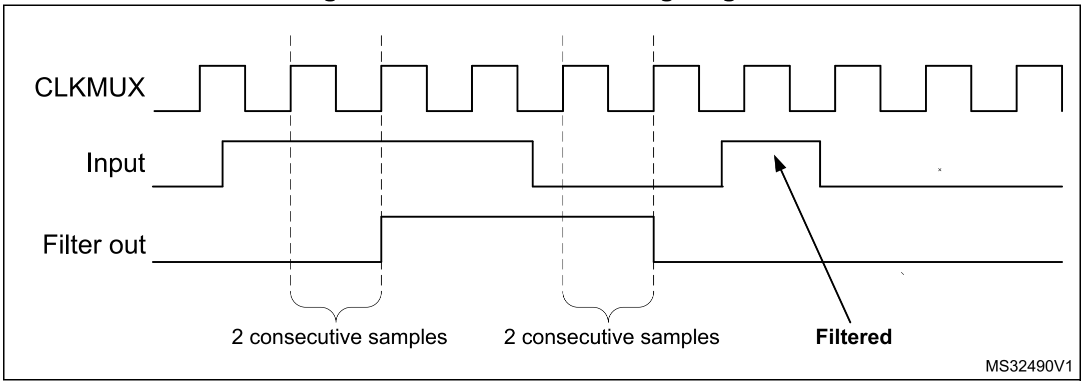
The digital filters are divided into two groups:

- The first group of digital filters protects the LPTIM external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that should be detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 496* shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

Figure 496. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

43.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

Table 311. Prescaler division ratios

programming	dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

43.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it will be ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled will be discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

43.4.8 Operating mode

The LPTIM features two operating modes:

- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when reaching the ARR value.

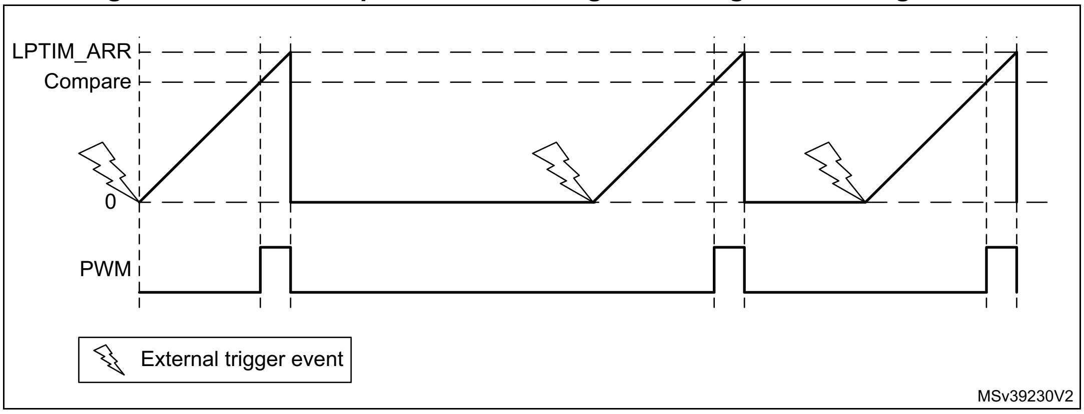
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event will re-start the timer. Any trigger event occurring after the counter starts and before the counter reaches ARR will be discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the counter register has stopped (contains zero value), will start the counter for a new one-shot counting cycle as shown in *Figure 497*.

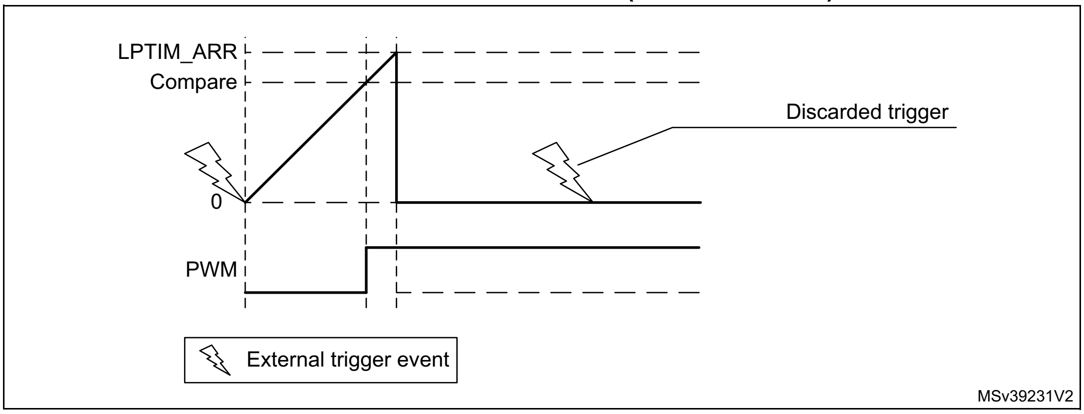
Figure 497. LPTIM output waveform, single counting mode configuration



Set-once mode activated:

It should be noted that when the WAVE bit-field in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 498*.

Figure 498. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = ‘00’), the SNGSTRT setting will start the counter for one-shot counting.

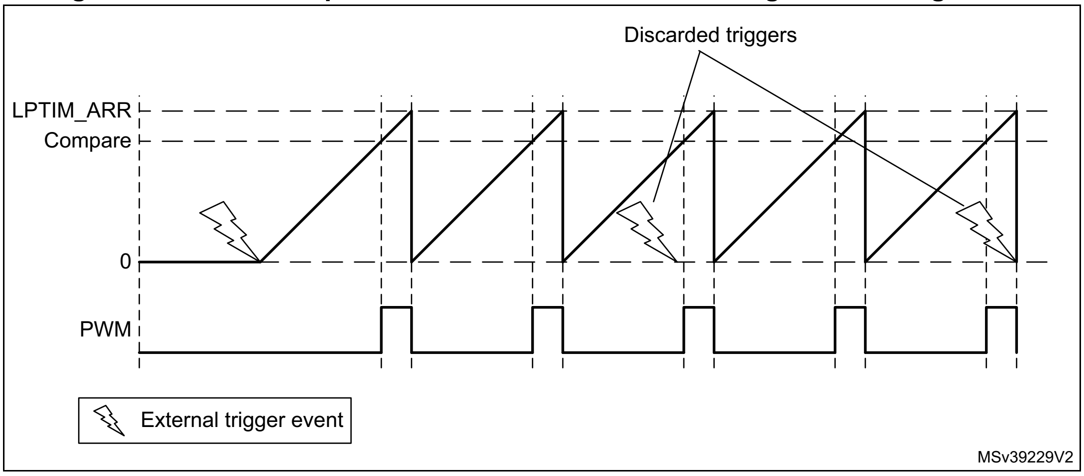
Continous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set will start the counter for continuous counting. Any subsequent external trigger event will be discarded as shown in *Figure 499*.

In case of software start (TRIGEN[1:0] = ‘00’), setting CNTSTRT will start the counter for continuous counting.

Figure 499. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT will switch the LPTIM to the One-shot mode. The counter (if active) will stop as soon as it reaches ARR.

If the One-shot mode was previously selected, setting CNTSTRT will switch the LPTIM to the Continuous mode. The counter (if active) will restart as soon as it reaches ARR.

43.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event will start the timer, any successive trigger event will reset the counter and the timer will restart.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

43.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CMP (compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CMP. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT registers.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CMP register value.

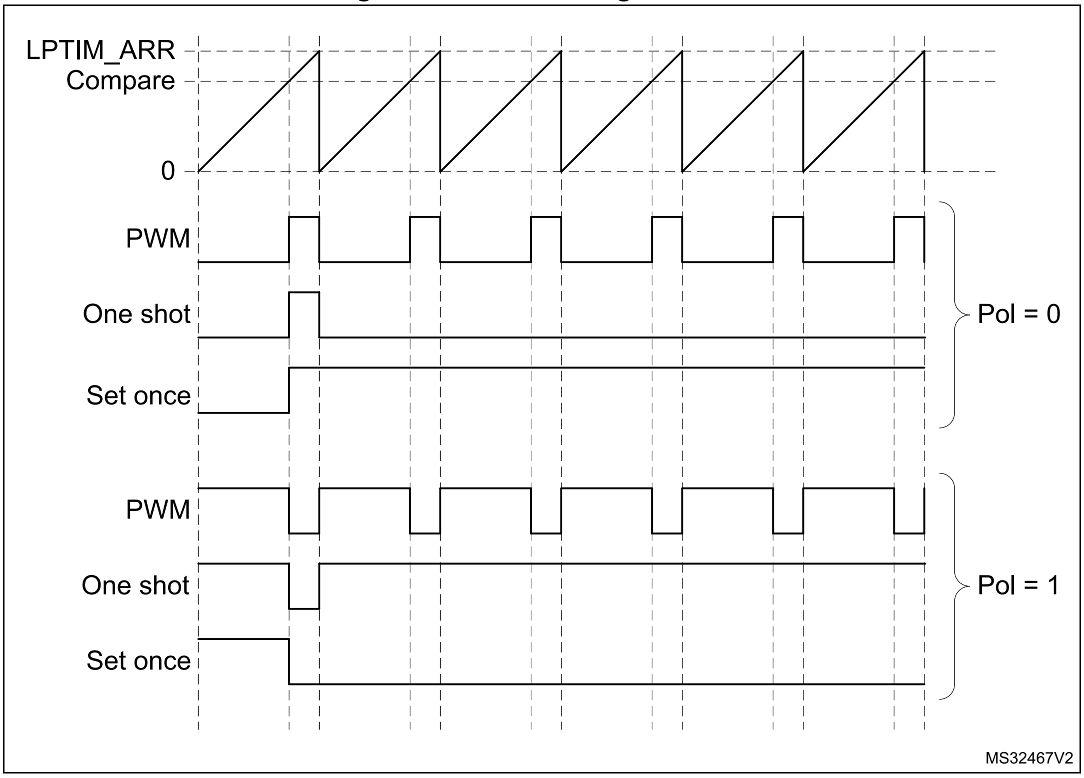
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to ‘0’ forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to ‘1’ forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value will change immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. *Figure 500* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL bit.

Figure 500. Waveform generation



43.4.11 Register update

The LPTIM_ARR register and LPTIM_CMP register are updated immediately after the APB bus write operation, or at the end of the current period if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR and the LPTIM_CMP registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR and the LPTIM_CMP registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR and the LPTIM_CMP registers are updated at the end of the current period, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag and the CMPOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register and the LPTIM_CMP register.

After a write to the LPTIM_ARR register or the LPTIM_CMP register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag or the CMPOK flag be set, will lead to unpredictable results.

43.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source will be used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal should never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

43.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR and LPTIM_IER registers must be modified only when the LPTIM is disabled.

43.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bit-field to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic will elapse before the reset is taken into account. This will make the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.
- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register will reset its content to zero. Asynchronous reset should be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset should be applied only when there is enough insurance that no toggle will occur on the LPTIM Input1.
  It should be noted that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: **There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So developer should make sure that these two mechanisms are used exclusively.**

43.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

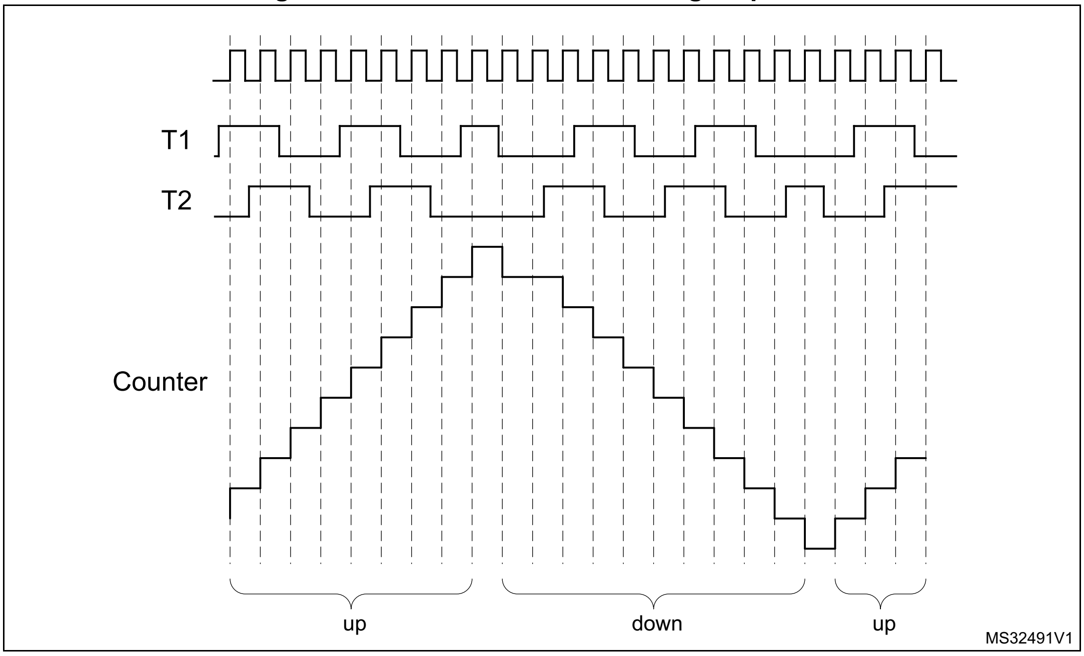
Table 312. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).

Figure 501. Encoder mode counting sequence



43.4.16 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the DBG_LPTIM_STOP configuration bit in the DBG module.

43.5 LPTIM low-power modes

Table 313. Effect of low-power modes on the LPTIM

Mode	Description
CSleep (MPU or MCU sub-system state)	No effect. LPTIM interrupts cause the sub-system to exit CSleep mode.
Stop + LP-Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode.
LPLV-Stop	Same as Stop + LP-Stop mode, except that only LSE or LSI are allowed as kernel clock source.
Standby or Shutdown	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

43.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_IER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).

Note: If any bit in the LPTIM_IER register (Interrupt Enable Register) is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.

Table 314. Interrupt events

Interrupt event	Description
Compare match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the compare register (LPTIM_CMP).
Auto-reload match	Interrupt flag is raised when the content of the Counter register (LPTIM_CNT) matches the content of the Auto-reload register (LPTIM_ARR).
External trigger event	Interrupt flag is raised when an external trigger event is detected
Auto-reload register update OK	Interrupt flag is raised when the write operation to the LPTIM_ARR register is complete.
Compare register update OK	Interrupt flag is raised when the write operation to the LPTIM_CMP register is complete.
Direction change	Used in Encoder mode. Two interrupt flags are embedded to signal direction change: – UP flag signals up-counting direction change – DOWN flag signals down-counting direction change.

43.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

43.7.1 LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN	UP	ARR OK	CMP OK	EXT TRIG	ARRM	CMPM
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMPOK**: Compare register update OK

CMPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_CMP register has been successfully completed. CMPOK flag can be cleared by writing 1 to the CMPOKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CMPM**: Compare match

The CMPM bit is set by hardware to inform application that LPTIM_CNT register value reached the LPTIM_CMP register's value. CMPM flag can be cleared by writing 1 to the CMPMCF bit in the LPTIM_ICR register.

43.7.2 LPTIM interrupt clear register (LPTIM_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN CF	UPCF	ARRO KCF	CMPO KCF	EXTTR IGCF	ARRM CF	CMPM CF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag

Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMPOKCF**: Compare register update OK clear flag

Writing 1 to this bit clears the CMPOK flag in the LPTIM_ISR register

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag

Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag

Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CMPMCF**: Compare match clear flag

Writing 1 to this bit clears the CMPM flag in the LPTIM_ISR register

43.7.3 LPTIM interrupt enable register (LPTIM_IER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNI E	UPIE	ARRO KIE	CMPO KIE	EXT TRIGIE	ARRM IE	CMPM IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable

0: DOWN interrupt disabled

1: DOWN interrupt enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable

0: UP interrupt disabled

1: UP interrupt enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable

0: ARROK interrupt disabled

1: ARROK interrupt enabled

Bit 3 **CMPOKIE**: Compare register update OK Interrupt Enable

0: CMPOK interrupt disabled

1: CMPOK interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable

0: EXTTRIG interrupt disabled

1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable

0: ARRM interrupt disabled

1: ARRM interrupt enabled

Bit 0 **CMPMIE**: Compare match Interrupt Enable

0: CMPM interrupt disabled

1: CMPM interrupt enabled

Caution: The LPTIM_IER register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’)

43.7.4 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRELOAD	WAVPOL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode

0: Encoder mode disabled

1: Encoder mode enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Please refer to Section 43.3: LPTIM implementation.

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:

0: the counter is incremented following each internal clock pulse

1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM_ARR and the LPTIM_CMP registers update modality

0: Registers are updated after each APB bus write access

1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVEPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CMP registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CMP registers

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode

1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started will be ignored

1: A trigger event arriving when the timer is already started will reset and restart the counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: software trigger (counting start is initiated by software)

01: rising edge is the active edge

10: falling edge is the active edge

11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that will serve as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 43.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any trigger active level change is considered as a valid trigger
01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that should be detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: any external clock signal level change is considered as a valid transition

01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

If LPTIM is clocked by an external clock source:

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00:the rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 1 is active.

01:the falling edge is the active edge used for counting
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 2 is active.

10:both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the encoder sub-mode 3 is active.

11:not allowed

Refer to *Section 43.4.15: Encoder mode* for more details about Encoder mode sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM will use:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

43.7.5 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST ARE	COUN TRST	CNT STRT	SNG STRT	ENA BLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register will asynchronously reset LPTIM_CNT register content.

This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit will trigger a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software should consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer will not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = ‘00’), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than ‘00’), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM will stop at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It will be automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0:LPTIM is disabled

1:LPTIM is enabled

43.7.6 LPTIM compare register (LPTIM_CMP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CMP[15:0]:** Compare value
CMP is the compare value used by the LPTIM.

Caution: The LPTIM_CMP register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

43.7.7 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CMP[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

43.7.8 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value

When the LPTIM is running with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. So in this case it is necessary to perform two consecutive read accesses and verify that the two returned values are identical.

It should be noted that for a reliable LPTIM_CNT register read access, two consecutive read accesses must be performed and compared. A read access can be considered reliable when the values of the two consecutive read accesses are equal.

43.7.9 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]:** LPTIM input 2 selection

The IN2SEL bits control the LPTIM Input 2 multiplexer, which connect LPTIM Input 2 to one of the available inputs.

00: lptim_in2_mux0

01: lptim_in2_mux1

10: lptim_in2_mux2

11: lptim_in2_mux3

For connection details refer to *Section 43.4.3: LPTIM input and trigger mapping*.

Note: If the LPTIM does not support encoder mode feature, these bits are reserved. Please refer to Section 43.3: LPTIM implementation.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]:** LPTIM input 1 selection

The IN1SEL bits control the LPTIM Input 1 multiplexer, which connects LPTIM Input 1 to one of the available inputs.

00: lptim_in1_mux0

01: lptim_in1_mux1

10: lptim_in1_mux2

11: lptim_in1_mux3

For connection details refer to *Section 43.4.3: LPTIM input and trigger mapping*.

Note: If LPTIM does not implement external input clock, these bits are reserved. Please refer to Section 43.3: LPTIM implementation.

Caution: The LPTIM_CFGR2 register must only be modified when the LPTIM is disabled (ENABLE bit reset to ‘0’).

43.7.10 LPTIM x peripheral hardware configuration register (LPTIMx_HWCFGR)(x = 1 to 5)

Address offset: 0x3F0

Reset value: Block 1: 0x0001 0804

Reset value: Block 2: 0x0001 0804

Reset value: Block 3: 0x0000 0804

Reset value: Block 4: 0x0000 0804

Reset value: Block 5: 0x0000 0804

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG4[7:0]								Res.	Res.	Res.	Res.	CFG3[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG2[7:0]								CFG1[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **CFG4[7:0]**: peripheral hardware configuration 4

This is read only bit-field and is read as CFG4[7:0] = option register

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CFG3[3:0]**: peripheral hardware configuration 3

This is read only bit-field and is read as CFG3[3:0] = encoder mode(1:enabled and 0:disabled)

Bits 15:8 **CFG2[7:0]**: peripheral hardware configuration 2

This is read only bit-field and is read as CFG2[7:0] = trigger number

Bits 7:0 **CFG1[7:0]**: peripheral hardware configuration 1

This is read only bit-field and is read as CFG1[7:0] = input multiplexer size

43.7.11 LPTIM peripheral version identification register (LPTIM_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[7:4]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[7:4]**: Major revision
This is read only bit-field and is read as MAJREV[7:4] = 1.

Bits 3:0 **MINREV[3:0]**: Minor revision
This is a read only bit-field and is read as MINREV[3:0] = 4.

43.7.12 LPTIM peripheral type identification register (LPTIM_PIDR)

Address offset: 0x3F8

Reset value: 0x0012 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **P_ID[31:0]**: Peripheral type identifier
This a read only register and is always read as P_ID[31:0] = 0x00120011.

43.7.13 LPTIM registers map size identification register (LPTIM_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
S_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **S_ID[31:0]**: Registers map size identifier

This a read only register and is always read as S_ID[31:0] = 0xA3C5DD01.

This register identifies the size of the memory region allocated to the LPTIM registers. The size of this memory region is 1 KB.

43.7.14 LPTIM register map

The following table summarizes the LPTIM registers.

Table 315. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWN(1)	UP(1)	ARROK	CMPOK	EXTTRIG	ARRM	CMPM
	Reset value																										0	0	0	0	0	0	0
0x004	LPTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNCF(1)	UPCF(1)	ARROKCF	CMPOKCF	EXTTRIGCF	ARRMCF	CMPMCF
	Reset value																										0	0	0	0	0	0	0
0x008	LPTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOWNIE(1)	UPIE(1)	ARROKIE	CMPOKIE	EXTTRIGIE	ARRMIE	CMPMIE
	Reset value																										0	0	0	0	0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(1)	COUNTMODE	PRELOAD	WAVEPOL	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]			Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0		0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value																											0	0			0	0
0x3F0	LPTIM_ HWCFGR	CFG4[27:20]								Res.	Res.	Res.	Res.	CFG3 [19:16]				CFG2[15:8]								CFG1[7:0]							
	Reset value	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x3F4	LPTIM_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [7:4]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0x3F8	LPTIM_PIDR	P_ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0x3FC	LPTIM_SIDR	S_ID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

1. If LPTIM does not support encoder mode feature, this bit is reserved. Please refer to *Section 43.3: LPTIM implementation*.

Refer to *Section 2.5 on page 153* for the register boundary addresses.

44 System timer generator (STGEN)

44.1 STGEN introduction

The STGEN generates a time-count value that provides a consistent view of time for multiple processors and other blocks in a device.

44.2 STGEN main features

- 64-bit wide to avoid roll-over issues
- Starts from 0 or a programmable value
- Control APB interface that enables the timer to be saved and restored across power-down events
- Read-only APB interface that enables the timer value to be read by non-secure software and debug tools
- hltdbg input to stop the timer value incrementing during full-system debug

44.3 STGEN functional description

The Cortex®-A7 generic timer schedules events and triggers interrupts based on an incrementing system counter (STGEN) value.

The generic timer provides:

- generation of timer events as interrupt outputs
- generation of event streams
- support for virtualization extensions

The Cortex®-A7 generic timer is compliant with the following Arm architecture reference manual system counter requirements:

- width: at least 56-bit wide
- frequency: increments at a fixed frequency

  The STGEN clock is simply derived from the external oscillator (HSE) in most circumstances, with the following exceptions:
  - During boot phase, the STGEN is clocked by HSI (64 MHz) until HSE is set up.
  - During low-power modes (Stop with HSE off and Standby), the STGEN is stopped by software. It is enabled after returning from these low-power modes, with time adjustment by secure software according to the RTC.
- accuracy: the Arm architecture reference manual does not specify a required accuracy, but recommends that the counter does not gain or lose more than ten seconds in a 24-hour period. This requirement is supported by software with occasional STGEN update (by increment) according to the RTC.

  The use of a lower-frequency mode must not affect the implemented accuracy.
- start-up: the counter starts from 0 at POR.

The STGEN provides a uniform view of system time.

Note: Arm recommends that the system counter is in an always-on power domain, which is not supported in the current implementation. The STGEN must then be saved and restored before Standby mode entry, and restored at Standby exit by secure software.

During low-power modes, the counter may increment by larger steps using a lower frequency. In this case, the counter must support transitions between high-frequency high-precision operation, and lower-frequency lower-precision operation, without any impact on the required counter accuracy.

The software can access the STGENC_CNTFIDO register to read the clock frequency of the system counter, and to modify the value of this register.

44.3.1 STGEN block diagram

The figure below shows the STGEN overview, with:

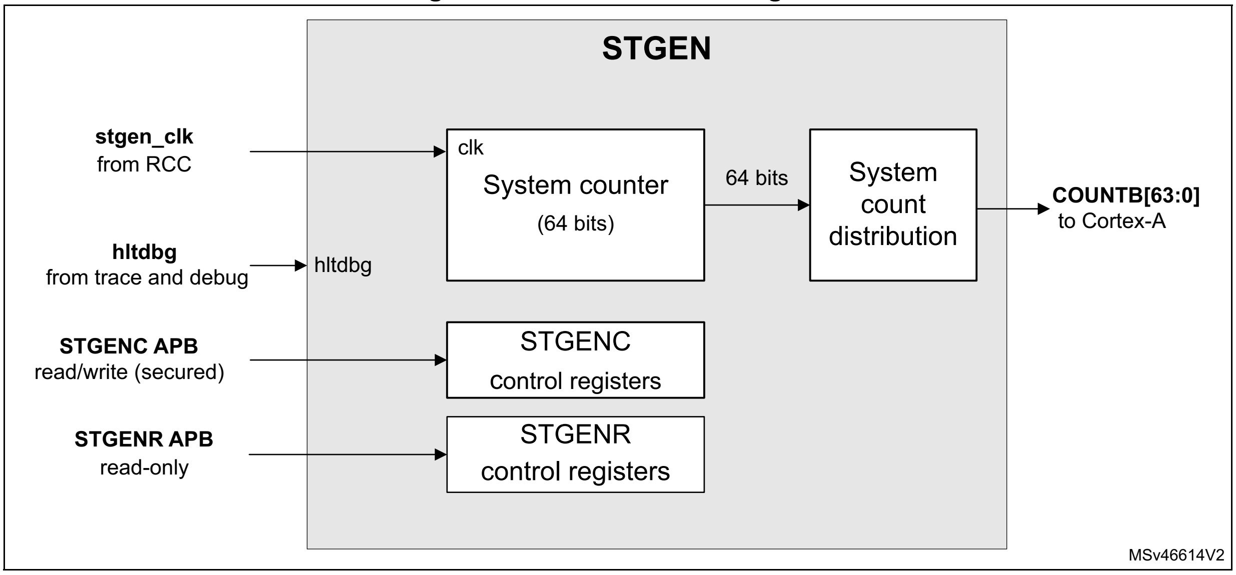
- a 64-bit system counter
- control registers with two APB ports, corresponding to STGENC and STGENR registers
- system count distribution logic to format the system counter value to Cortex®-A7 with gray encoding and frequency interpolation.

Note: The two APB ports are used because of TrustZone security isolation as follows:

- STGENC is read/write only by Cortex®-A7 secure.

- STGENR is read only by secure or non-secure.

Figure 502. STGEN block diagram



Note: hltdbg is a combinatorial AND of DBGACK outputs from Cortex®-A7 processors, so that the STGEN is halted when both Cortex®-A7 processors are in debug.

44.4 STGEN registers

STGEN control registers are organized in two sets accessible by two APB interfaces:

- The first set accessible as STGEN**C** APB is read/write and made TrustZone secured.
- The second set accessible by STGEN**R** APB is read-only and not concerned by TrustZone security.

44.4.1 STGENC control register (STGENC_CNTCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HLTDBG	EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HLTDBG:** Halt on debug
0: Do not halt on debug; hltdbg signal into the counter has no effect.
1: Halt on debug; when hltdbg is driven high, the count value is held static.

Bit 0 **EN:** Enable
0: Counter disabled and not incrementing
1: Counter enabled and incrementing

44.4.2 STGENC status register (STGENC_CNTSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBGH:** Debug halted
0: Normal operation
1: Counter halted due to system-wide debug state

Bit 0 Reserved, must be kept at reset value.

44.4.3 STGENC value lower register (STGENC_CNTCVL)

Address offset: 0x008

Reset value: 0x0000 0000

Note: The control interface must clear the STGENC_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVL_L_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNTCVL_L_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVL_L_32[31:0]**: Current value of the timestamp counter, lower 32 bits

To change the current timestamp value, write the lower 32 bits of the new value to this register before writing the upper 32 bits to STGENC_CNTCVU. The timestamp value is not changed until the STGENC_CNTCVU register is written to.

44.4.4 STGENC value upper register (STGENC_CNTCVU)

Address offset: 0x00C

Reset value: 0x0000 0000

Note: The control interface must clear the STGENC_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_U_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNTCVU_U_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU_U_32[31:0]**: Current value of the timestamp counter, upper 32 bits

To change the current timestamp value, write the lower 32 bits of the new value to STGENC_CNTCVL before writing the upper 32 bits to this register. The 64-bit timestamp value is updated with the value from both writes when this register is written to.

44.4.5 STGENC base frequency register (STGENC_CNTFID0)

Address offset: 0x020

Reset value: 0x0000 0000

Note: The control interface must clear the STGEN_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]:** Frequency in number of ticks per second
It can be specified up to 4 GHz. This register must be programmed to match the clock frequency of the clock input to TSGEN (tsgen_clk), in ticks per second. For example, for a 50 MHz clock, the register value has to be 0x02FAF080.

44.4.6 STGENC peripheral ID4 register (STGENC_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]:** Always 0b0000
Indicates that the device only occupies 4 Kbytes of memory.

Bits 3:0 **DES_2[3:0]:** Part of designer identification
Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.
0b0100 JEDEC continuation code

44.4.7 STGENC peripheral ID5 register (STGENC_PIDR5)

Address offset: 0xFD4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR5[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR5[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR5[31:0]:** Peripheral ID5

44.4.8 STGENC peripheral ID6 register (STGENC_PIDR6)

Address offset: 0xFD8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR6[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR6[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR6[31:0]:** Peripheral ID6

44.4.9 STGENC peripheral ID7 register (STGENC_PIDR7)

Address offset: 0xFDC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR7[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR7[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR7[31:0]:** Peripheral ID7

44.4.10 STGENC peripheral ID0 register (STGENC_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PART_0[7:0]:** Bits[7:0] of the 12-bit part number of the component
The designer of the component assigns this part number.

44.4.11 STGENC peripheral ID1 register (STGENC_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DES_0[3:0]:** Part of designer identification
Together PIDR1.DES_0, PIDR2.DES_1, and PIDR4.DES_2 identify the designer of the component.

Bits 3:0 **PART_1[3:0]:** Bits[11:8] of the 12-bit part number of the component
The designer of the component assigns this part number.

44.4.12 STGENC peripheral ID2 register (STGENC_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** device revision number

Bit 3 **JEDEC:** always 1
Indicates that the JEDEC-assigned designer ID is used.

Bits 2:0 **DES_1[2:0]:** part of designer identification
Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

44.4.13 STGENC peripheral ID3 register (STGENC_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Customer version
0b0000 indicates there are no errata fixes to this component.

Bits 3:0 **CMOD[3:0]:** Customer modified
Indicates whether the customer has modified the behavior of the component. In most cases, this bitfield is 0b0000 (no modification done). Customers change this value when they make authorized modifications to this component.

44.4.14 STGENC component ID0 register (STGENC_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_0[7:0]:** Preamble 0
Contains bits[7:0] of the component identification code.

44.4.15 STGENC component ID1 register (STGENC_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Class of the component
For example, the component can be a ROM table or a generic CoreSight™ SoC-400 component. Contains bits[15:12] of the component identification code.
0b1111 indicates the component is a CoreSight™ SoC-400 component.

Bits 3:0 **PRMBL_1[3:0]:** preamble 1
Contains bits[11:8] of the component identification code.

44.4.16 STGENC component ID2 register (STGENC_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_2[7:0]:** Preamble 2
Contains bits[23:16] of the component identification code.

44.4.17 STGENC component ID3 register (STGENC_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_3[7:0]:** Preamble 3
Contains bits[31:24] of the component identification code.

44.4.18 STGENR value lower register (STGENR_CNTCVL)

Address offset: 0x000

Reset value: 0x0000 0000

Note: The control interface must clear the STGEN_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVL_L_32[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNTCVL_L_32[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CNTCVL_L_32[31:0]**: Current value of the timestamp counter (the lower 32 bits)

44.4.19 STGENR value upper register (STGENR_CNTCVU)

Address offset: 0x004

Reset value: 0x0000 0000

Note: The control interface must clear the STGEN_CNTCR.EN bit before writing to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_U_32[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNTCVU_U_32[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CNTCVU_U_32[31:0]**: Current value of the timestamp counter (the upper 32 bits)

44.4.20 STGENR peripheral ID4 register (STGENR_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]:** Always 0b0000

Indicates that the device only occupies 4 Kbytes of memory.

Bits 3:0 **DES_2[3:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

44.4.21 STGENR peripheral ID5 register (STGENR_PIDR5)

Address offset: 0xFD4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR5[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIDR5[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR5[31:0]:** Peripheral ID5

44.4.22 STGENR peripheral ID6 register (STGENR_PIDR6)

Address offset: 0xFD8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR6[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIDR6[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR6[31:0]:** Peripheral ID6

44.4.23 STGENR peripheral ID7 register (STGENR_PIDR7)

Address offset: 0xFDC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIDR7[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIDR7[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PIDR7[31:0]:** Peripheral ID7

44.4.24 STGENR peripheral ID0 register (STGENR_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PART_0[7:0]:** Bits[7:0] of the 12-bit part number of the component

44.4.25 STGENR peripheral ID1 register (STGENR_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DES_0[3:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2 identify the designer of the component.

Bits 3:0 **PART_1[3:0]:** Bits[11:8] of the 12-bit part number of the component

44.4.26 STGENR peripheral ID2 register (STGENR_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]:** Device revision number

Bit 3 **JEDEC:** Always 1

Indicates that the JEDEC-assigned designer ID is used.

Bits 2:0 **DES_1[2:0]:** Part of designer identification

Together STGEN_PIDR1.DES_0, STGEN_PIDR2.DES_1, and STGEN_PIDR4.DES_2, identify the designer of the component.

44.4.27 STGENR peripheral ID3 register (STGENR_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]:** Errata fix identification

0b0000 indicates there are no errata fixes to this component.

Bits 3:0 **CMOD[3:0]:** Customer modified

Indicates whether the customer has modified the behavior of the component. In most cases, this bitfield is 0b0000 (no modification done). Customers change this value when they make authorized modifications to this component.

44.4.28 STGENR component ID0 register (STGENR_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_0[7:0]:** Preamble 0
Contains bits[7:0] of the component identification code.

44.4.29 STGENR component ID1 register (STGENR_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]:** Class of the component
Contains bits[15:12] of the component identification code.

Bits 3:0 **PRMBL_1[3:0]:** Preamble 1
Contains bits[11:8] of the component identification code.

44.4.30 STGENR component ID2 register (STGENR_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_2[7:0]:** Preamble 2
Contains bits[23:16] of the component identification code.

44.4.31 STGENR component ID3 register (STGENR_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_3[7:0]:** Preamble 3
Contains bits[31:24] of the component identification code.

44.4.32 STGEN register map

Table 316. STGENC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	STGENC_CNTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HLTDBG	EN
	Reset value																															0	0
0x004	STGENC_CNTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
	Reset value																															0	
0x008	STGENC_CNTCVL	CNTCVL_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	STGENC_CNTCVU	CNTCVU_U_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0100-0x01C	Reserved	Reserved																															
0x0020	STGENC_CNTFID0	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0x0FCC	Reserved	Reserved																															
0xFD0	STGENC_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 + 0x4*(x-5) (x=5 to 7)	STGENC_PIDRx	PIDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0	STGENC_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
	Reset value																									0	0	0	0	0	0	0	1

Table 316. STGENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	STGENC_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
	Reset value																									1	0	1	1	0	0	0	1
0xFE8	STGENC_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	DES_1 [2:0]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	STGENC_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	STGENC_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	STGENC_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1 [3:0]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	STGENC_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	STGENC_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

Table 317. STGENR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	STGENR_CNTCVL	CNTCVL_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	STGENR_CNTCVU	CNTCVU_U_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008-0xFCC	Reserved	Reserved																															
0xFD0	STGENR_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 + 0x4*(x-5) (x=5 to 7)	STGENR_PIDR5x	PIDRx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFE0-0xFDC	Reserved	Reserved																															
0xFE0	STGENR_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	STGENR_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
	Reset value																									1	0	1	1	0	0	0	1
0xFE8	STGENR_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	DES_1 [2:0]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	STGENR_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	STGENR_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	STGENR_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1 [3:0]			
	Reset value																									1	1	1	1	0	0	0	0

Table 317. STGENR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	STGENR_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
	Reset value																									0	1	0	1	0	0	0	0
0xFFC	STGENR_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

45 Watchdogs overview

The devices feature three embedded watchdog blocks which offer a combination of high safety, timing accuracy and flexibility.

45.1 Watchdog main features

- Two independent watchdogs (IWDG1 and IWDG2) dedicated to MPU.
  - The IWDG1 is TrustZone aware.
- One window watchdog (WWDG1) dedicated to the MCU.
- The MPU can receive an interrupt if the WWDG1 generates a reset, or an early interrupt.
- The MCU can receive the WWDG1 early interrupt.
- Watchdogs behavior can be configured via option bytes and debug options.
- The MPU can receive the IWDG1 or IWDG2 early interrupts.

45.2 Watchdog brief functional description

Both watchdog peripherals (Independent and Window) allow detecting and resolving malfunctions due to software or hardware failures.

The IWDG1 and IWDG2 are dedicated to MPU, while the WWDG1 is dedicated to MCU. In addition, the IWDG1 is located onto a secure bus (APB5), and can only be accessed by a secure application when TrustZone is enabled.

The *Figure 503* shows the interconnection of the watchdogs into the system.

The WWDG1 is clocked with the APB1 clock, and provides a reset and an early interrupt signal.

The early interrupt provided by the WWDG1 (wwdg1_it) is connected to both MPU (GIC) and MCU (NVIC) interrupt controllers. It allows the application to decide which processor has to handle emergency tasks if needed.

The WWDG1 also generates a reset to the MCU when a timeout occurs (wwdg1_out_rst signal). This signal is also routed to the EXTI in order to wake-up the MPU core if the MCU is reset by the WWDG1 block.

The independent watchdogs (IWDG1 and IWDG2) are clocked by the low-speed clock (LSI) and thus stay active even if the main clock fails. In this way they are best suited to applications which require the watchdog to run as a totally independent process outside the main application. The IWDGs are best suited to recover from unexpected software or hardware failures.

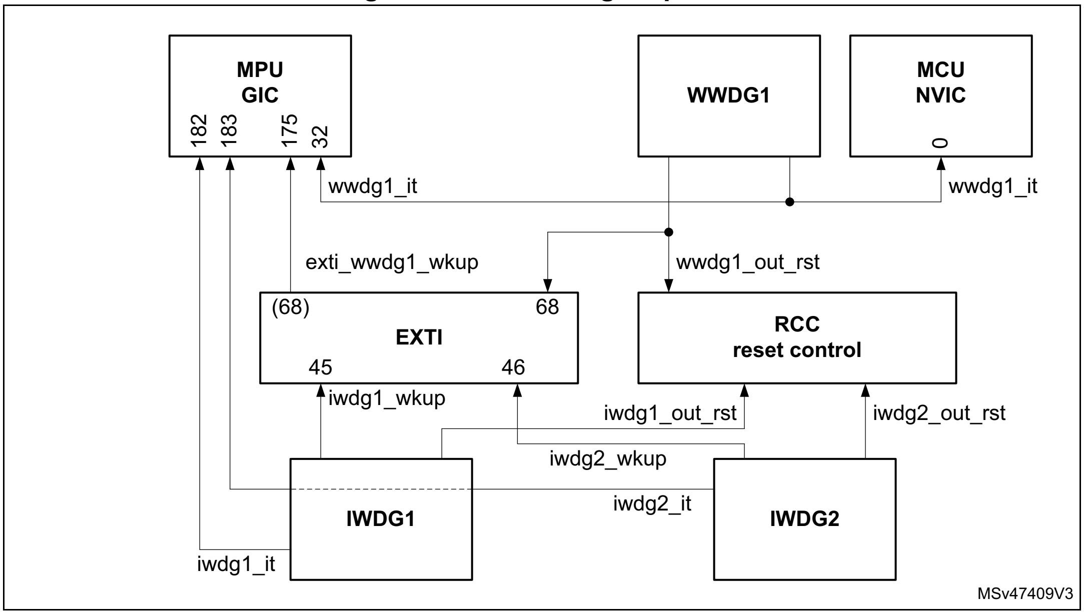
The early interrupt provided by the IWDG1 (iwdg1_it) and IWDG2 (iwdg2_it) is connected to the MPU (GIC) interrupt controller to handle emergency tasks if needed.

The early wake-up signal provided by the IWDG1 (iwdg1_wkup) and IWDG2 (iwdg2_wkup) is connected to the EXTI to wake-up the MPU if needed.

The IWDG1 and IWDG2 generate a system reset when a timeout occurs (iwdg[2:1]_out_rst signals). The IWDG1 and IWDG2 blocks are designed to work with the MPU, in addition, the IWDG1 is TrustZone aware.

Please refer to RCC, *Section : Clock distribution for watchdogs (IWDG[2:1], WWDG1)*, and *Section 10.3.5: Watchdog reset* for additional information.

Figure 503. Watchdogs top view



46 Independent watchdog (IWDG)

46.1 Introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 46.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

46.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

46.3 IWDG implementation

Table 318. STM32MP153x IWDG features [1]

IWDG modes/features	IWDG1	IWDG2
LSI used as IWDG kernel clock (iwdg_ker_ck)	X	X
Window function	X	X
Early wake-up interrupt generation	X	X
System reset generation [2]	X	X
Capability to work in system Stop	X	X
Capability to work in system Standby	X	X
Capability to generate an interrupt in system Stop	X	X
Capability to generate an interrupt in system Standby	-	-
Capability to be frozen when the MPU enters in debug mode	X [3]	X [4]
Capability to be frozen when the MCU enters in debug mode	-	-
Option bytes to control the activity in Stop mode [5]	X	X
Option bytes to control the activity in Standby mode [6]	X	X
Option bytes to control the hardware mode [7]	X	X

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.

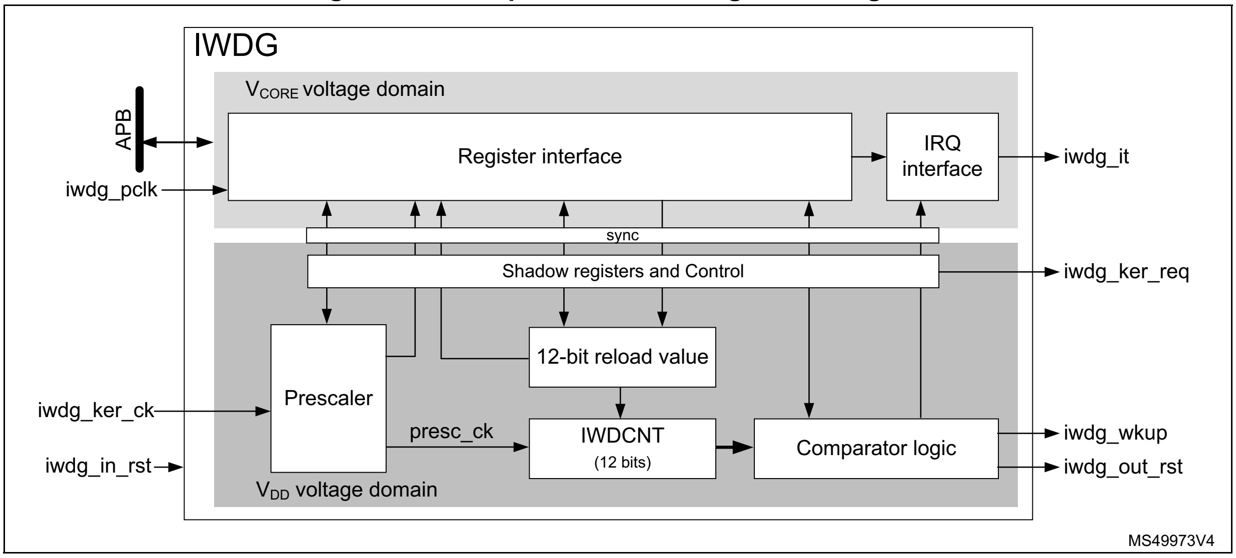
3. Controlled by bit IWDG1 of DBGMCU_APB5FZ1 register in the DBG block.
4. Controlled by bit IWDG2 of DBGMCU_APB4FZ1 register in the DBG block.
5. Controlled via option bytes OTP_IWDG1_FZ_STOP and OTP_IWDG2_FZ_STOP, respectively, for IWDG1 and IWDG2.
6. Controlled via option bytes OTP_IWDG1_FZ_STANDBY and OTP_IWDG2_FZ_STANDBY, respectively, for IWDG1 and IWDG2.
7. Controlled via option bytes OTP_IWDG1_HW and OTP_IWDG2_HW, respectively, for IWDG1 and IWDG2.

46.4 IWDG functional description

46.4.1 IWDG block diagram

Figure 504 shows the functional blocks of the independent watchdog module.

Figure 504. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 46.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 319. IWDG delays versus actions [(1)]

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

46.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 320*.

Table 320. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

46.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 46.3* for details.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

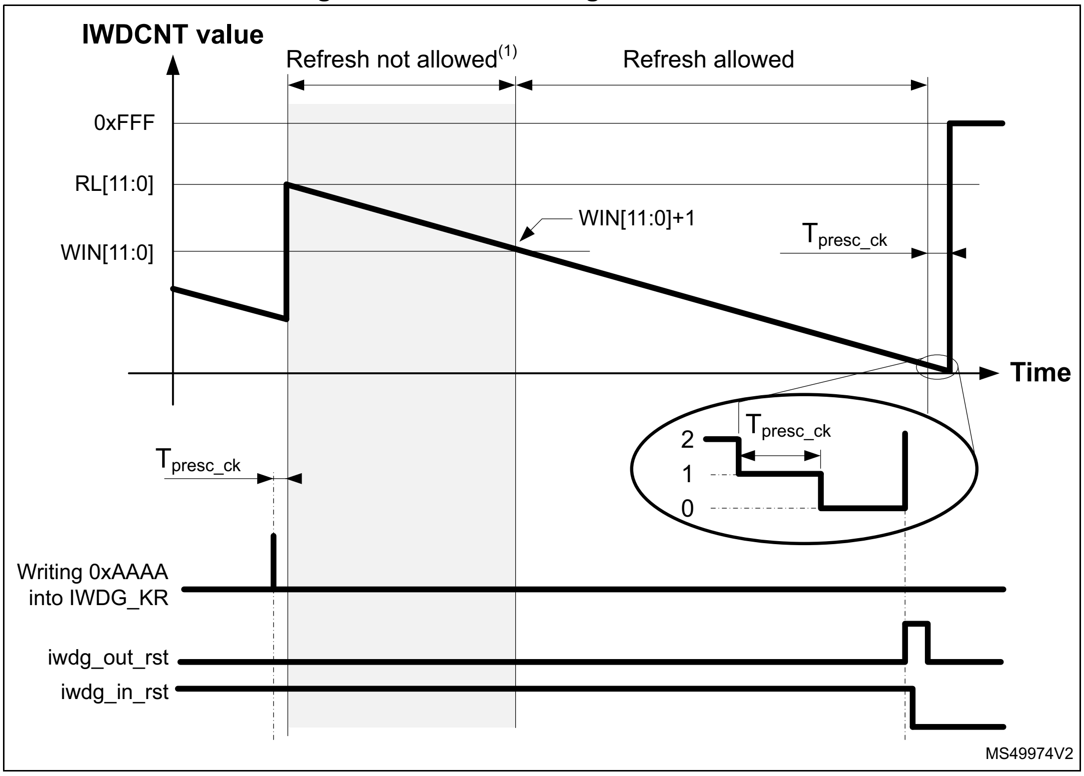
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 505*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 505. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

46.4.4 Window option

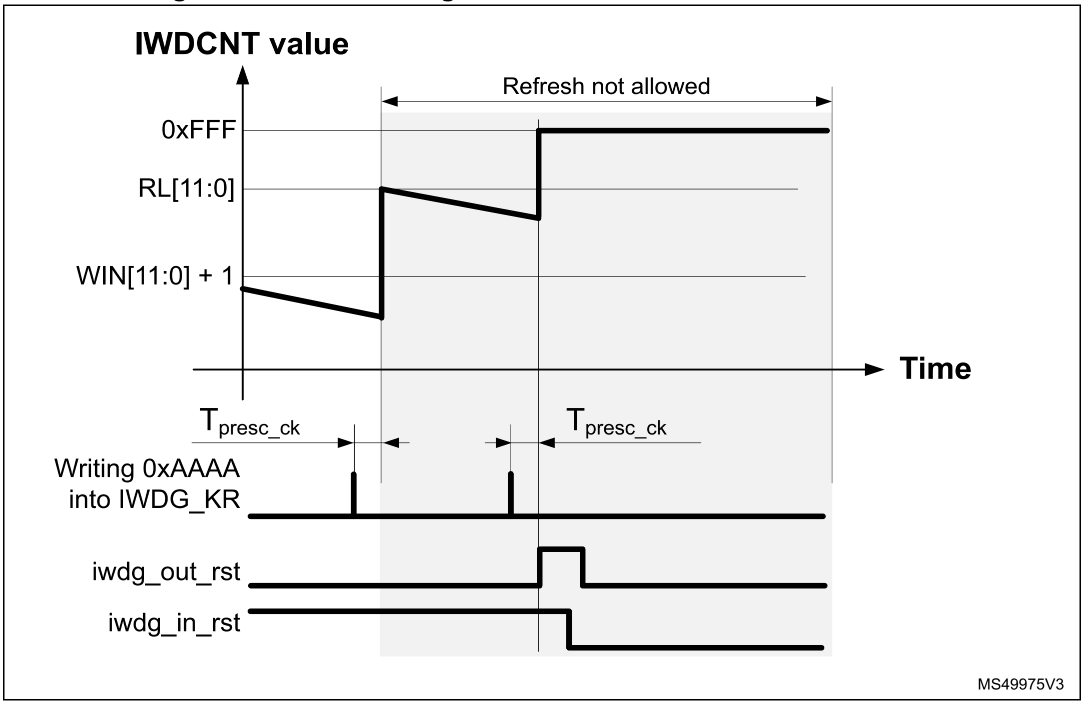
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 506*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 506. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

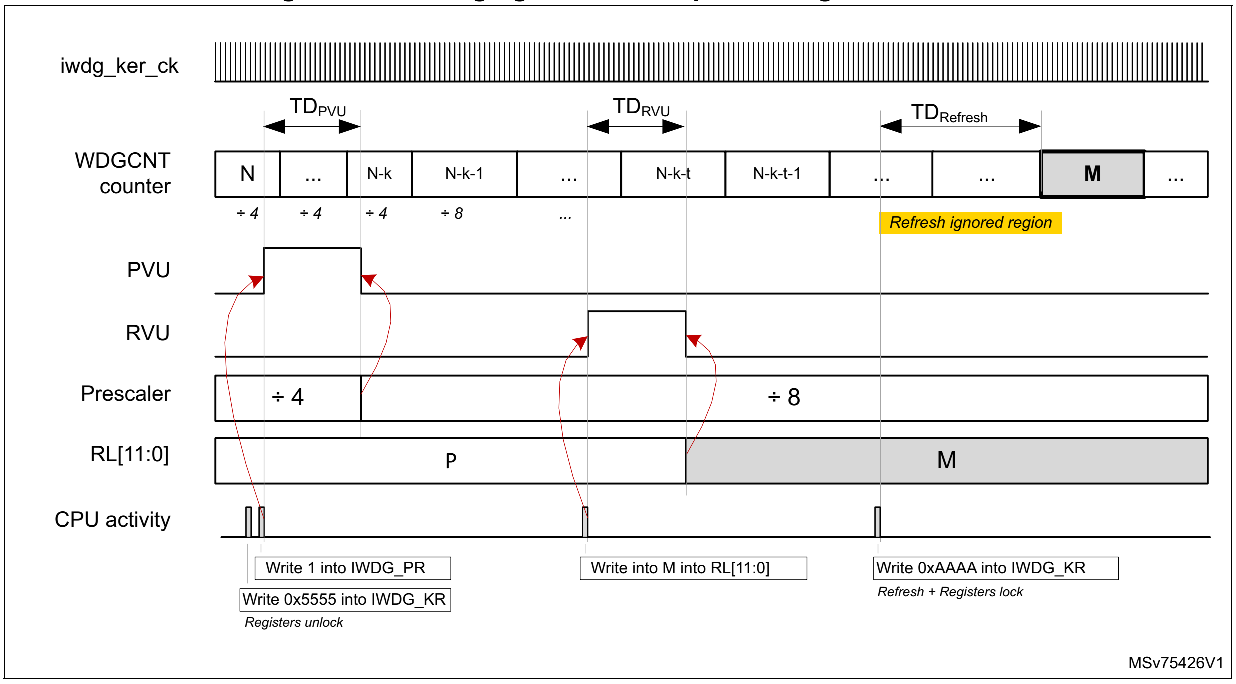
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 507. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 319: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

46.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 46.3* for details on the capabilities of this product.

46.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

46.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 46.3* for details.

Table 321. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 46.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 46.3* for details. IWDG interrupts do not make the device to exit from Standby mode.
Shutdown	The IWDG is not working.

46.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

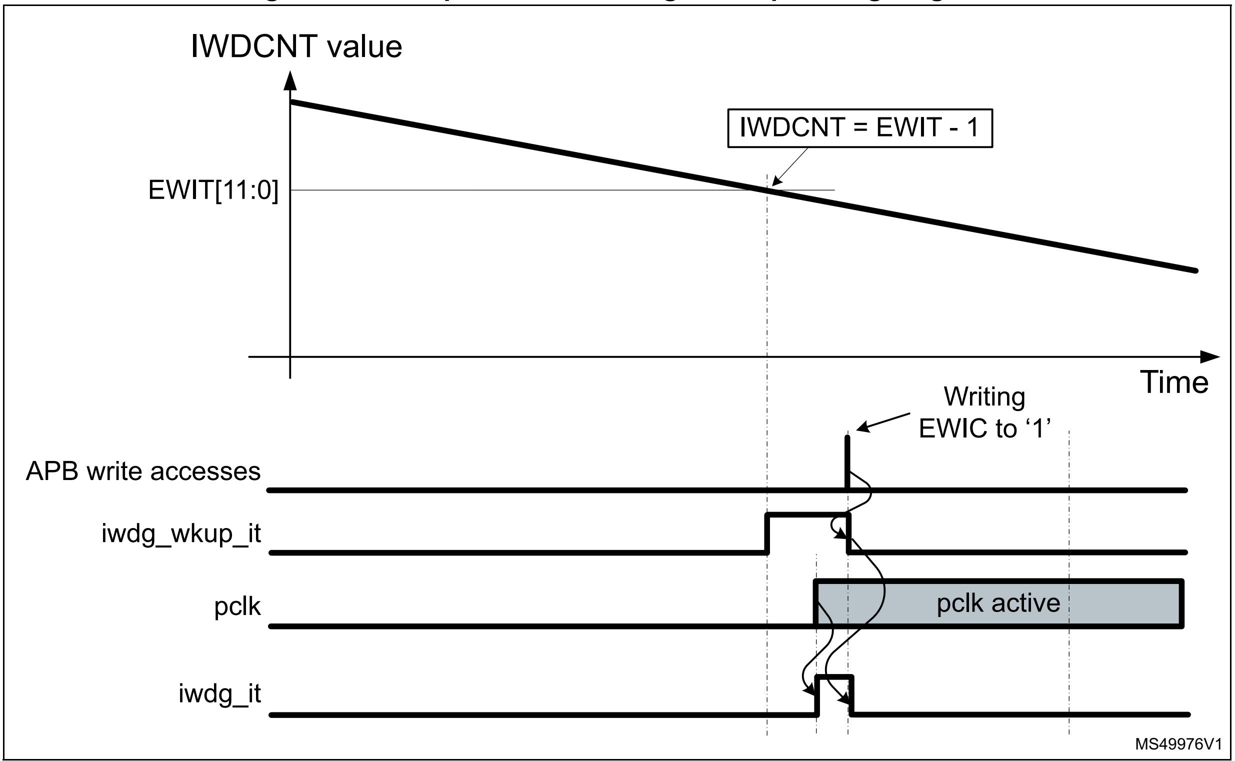
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG early wake-up interrupt register (IWDG_EWCR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 508. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.

2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Table 322 summarizes the IWDG interrupt request.

Table 322. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

46.7 IWDG registers

Refer to *Section 1.2 on page 115* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

46.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

46.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 46.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

46.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 46.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

46.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	r											r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 319: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 319: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 319: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 319: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

46.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 46.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

46.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EWIE	EWIC	Res.	Res.	EWIT[11:0]											
rw	w			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter window value

These bits are write access protected (see *Section 46.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

46.7.7 IWDG hardware configuration register (IWDG_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0081

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR_DEFAULT[3:0]				WINDOW[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **PR_DEFAULT[3:0]**: Prescaler default value

This bitfield gives the default value of the prescaler (see *PR[3:0]: Prescaler divider* of *IWDG prescaler register (IWDG_PR)*

0000: Prescaler (PR[3:0]) default value after reset is 0000'
0001: Prescaler (PR[3:0]) default value after reset is 0001'
0010: Prescaler (PR[3:0]) default value after reset is 0010'
0011: Prescaler (PR[3:0]) default value after reset is 0011'
0100: Prescaler (PR[3:0]) default value after reset is 0100'
0101: Prescaler (PR[3:0]) default value after reset is 0101'
0110: Prescaler (PR[3:0]) default value after reset is 0110'
0111: Prescaler (PR[3:0]) default value after reset is 0111'
1xxx: Prescaler (PR[3:0]) default value after reset is 1000

Bits 3:0 **WINDOW[3:0]**: Support of Window function

0000: Window option disabled
0001: Window option enabled
Others: Reserved

46.7.8 IWDG version register (IWDG_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0030

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: IWDG major revision

Bits 3:0 **MINREV[3:0]**: IWDG minor revision

46.7.9 IWDG identification register (IWDG_IDR)

Address offset: 0x3F8

Reset value: 0x0012 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IWDG identifier

46.7.10 IWDG size identification register (IWDG_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: IWDG size identifier

The address decoding range is 1 Kbyte.

46.7.11 IWDG register map

Table 323. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													1	1	1	1
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																		0											0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	EWIC	Res.	Res.	EWIT[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x3F0	**IWDG _HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR_DEFAULT [3:0]				WINDOW [3:0]			
	Reset value																									1	0	0	0	0	0	0	1
0x3F4	**IWDG_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	1	0	0	0	0
0x3F8	**IWDG_IDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x3FC	**IWDG_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

47 System window watchdog (WWDG)

47.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

47.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 510*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

47.3 WWDG functional description

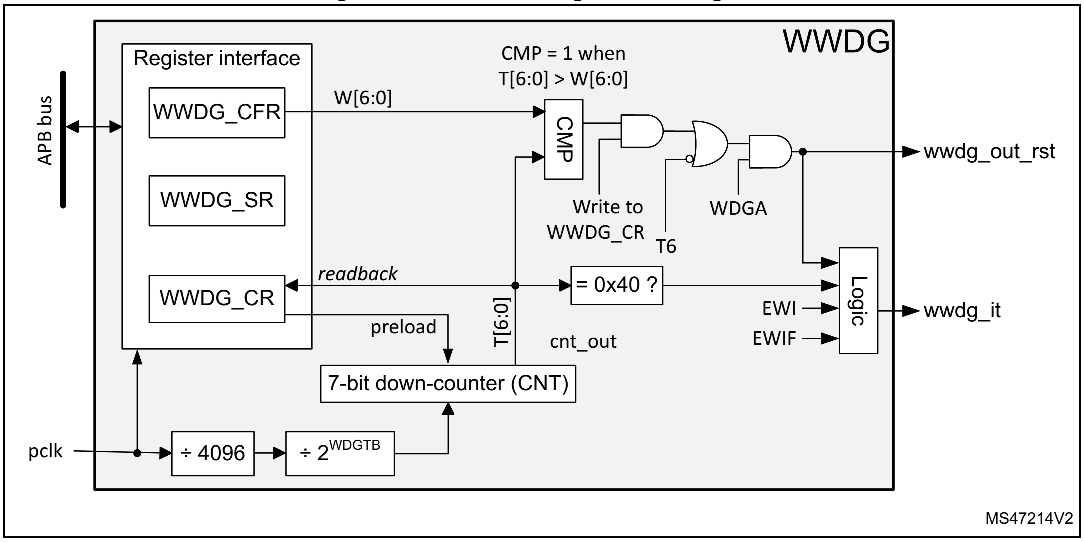
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 509* and to *Section 47.3.2: WWDG internal signals* for the WWDG block diagram.

47.3.1 WWDG block diagram

Figure 509. Watchdog block diagram



47.3.2 WWDG internal signals

Table 324 gives the list of WWDG internal signals.

Table 324. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

47.3.3 Enabling the watchdog

The watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again except by a reset.

47.3.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 510*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 510* describes the window watchdog process.

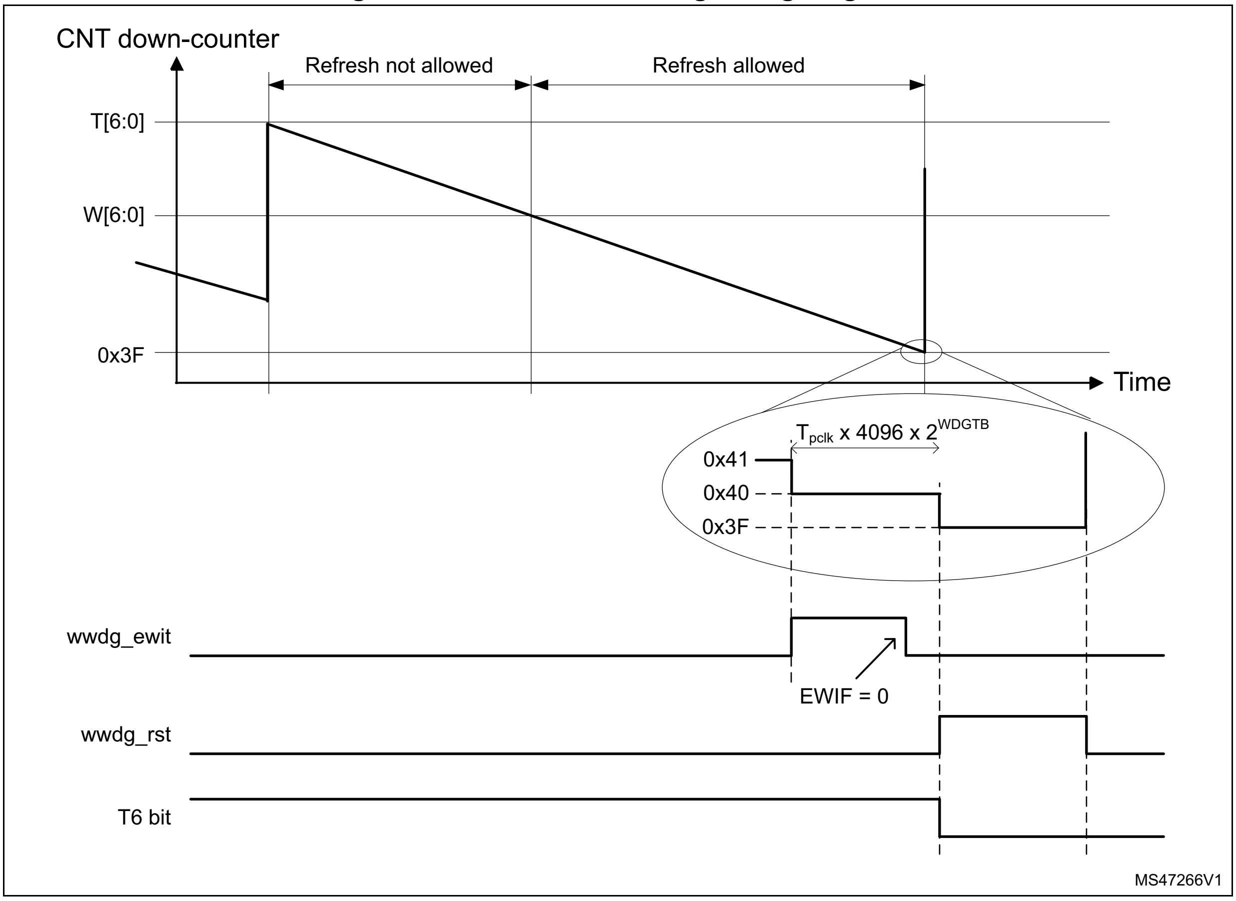
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

47.3.5 How to program the watchdog timeout

Use the formula in *Figure 510* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 510. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \qquad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69 ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

47.3.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 64: Debug support (DBG)*.

47.4 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

47.5 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

47.5.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096×2^{WDGTB}) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

47.5.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

47.5.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

47.5.4 WWDG hardware configuration register (WWDG_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PREDIV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PREDIV[15:0]**: The watchdog clock is prescaled by 4096 (PREDIV[15:0]).

47.5.5 WWDG version register (WWDG_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: WWDG major revision

Bits 3:0 **MINREV[3:0]**: WWDG minor revision

47.5.6 WWDG identification register (WWDG_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 0051

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: WWDG peripheral identifier

47.5.7 WWDG size identification register (WWDG_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: The address decoding range is 1 Kbyte

47.5.8 WWDG register map

The following table gives the WWDG register map and reset values.

Table 325. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0
0x00C-0x3EC	Reserved	Reserved																															
0x3F0	WWDG_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV[15:0]															
	Reset value																	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x3F4	WWDG_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	1	0	0	0	0	1
0x3F8	WWDG_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
0x3FC	WWDG_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

48 Real-time clock (RTC)

48.1 Introduction

The RTC provides an automatic wakeup to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

48.2 RTC main features

The RTC supports the following features (see *Figure 511: RTC block diagram*):

- Calendar with subsecond, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wakeup timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  – RTC fully securable
  – Alarm A, alarm B, wakeup Timer and timestamp individual secure or non-secure configuration

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the V_{BAT} pin.

The RTC clock sources can be:

- A 32.768 kHz external crystal (LSE)
- An external resonator or oscillator (LSE)
- The internal low power RC oscillator (LSI, with typical frequency of 32 kHz)
- The high-speed external clock (HSE), divided by a prescaler in the RCC.

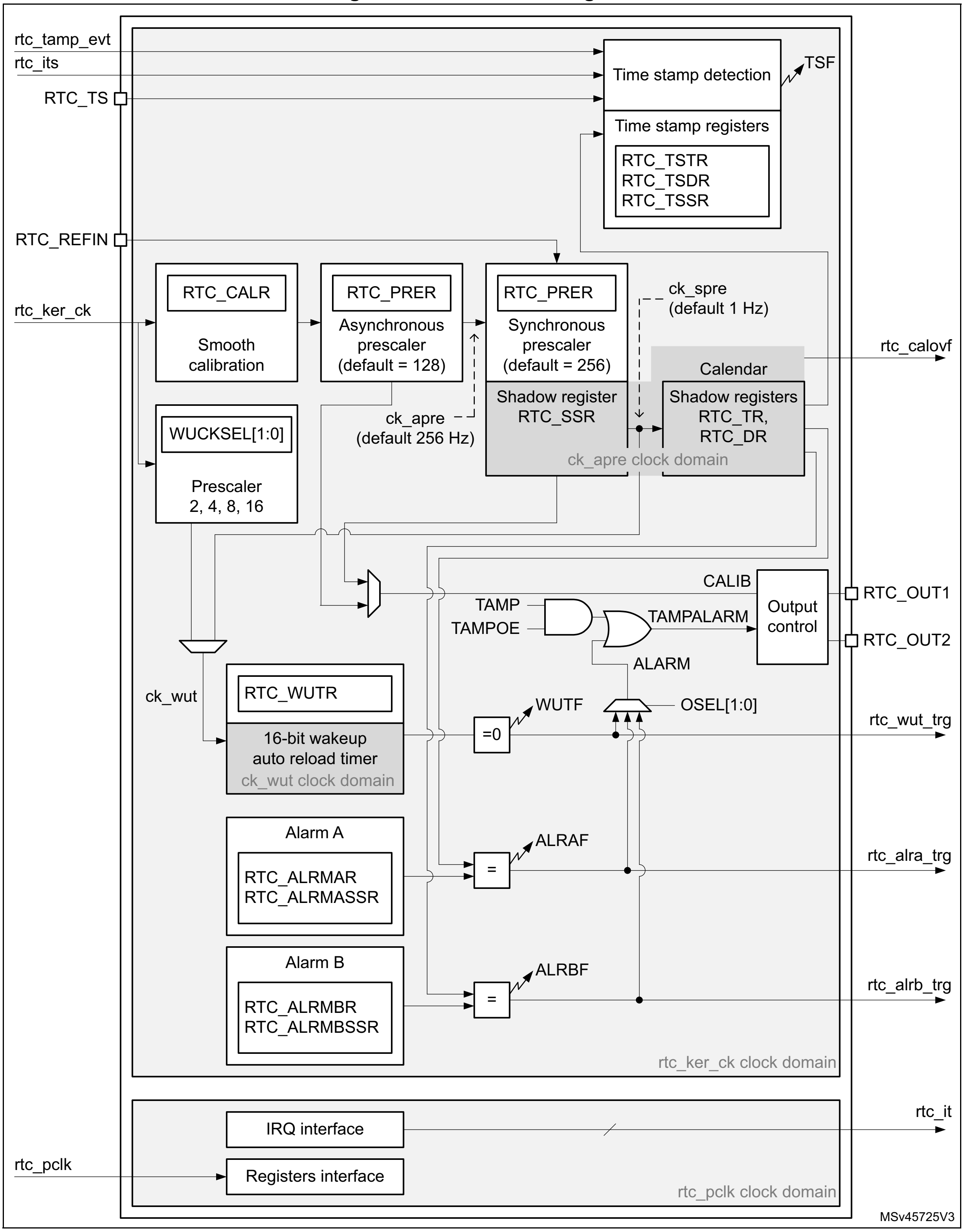
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE. When clocked by the LSI, the RTC is not functional in V_{BAT} mode, but is functional in all low-power modes.

All RTC events (Alarm, WakeUp Timer, Timestamp) can generate an interrupt and wakeup the device from the low-power modes.

48.3 RTC functional description

48.3.1 RTC block diagram

Figure 511. RTC block diagram



48.3.2 RTC pins and internal signals

Table 326. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2
RTC_LSCO	Output	RTC low-speed clock output

- RTC_OUT1 and RTC_OUT2 which selects one of the following two outputs:
  - CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
  - TAMPALRM: This output is the OR between TAMP and ALARM outputs.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wakeup outputs. TAMP is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 327. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_it	Output	RTC interrupts (refer to *Section 48.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wakeup timer event detection trigger
rtc_calovf	Output	RTC calendar overflow

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 48.4: RTC low-power modes* for more details.

Table 328. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

48.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

Both RTC and TAMP peripherals provide functions on these I/Os (refer to *Section 49: Tamper and backup registers (TAMP)*).

RTC_OUT1, RTC_LSCO, RTC_TS, TAMP_IN1, TAMP_OUT2 and TAMP_OUT3 are mapped on the same pin (PC13). RTC_LSCO, The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 329*.

Table 329. RTC pin PC13 configuration[(1)]

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E & TAMP2AM	TAMP3E & TAMP3AM & OUT3_RMP	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)	LSCOEN[1:0]
TAMPALRM output Push-Pull	01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care	Don't care	Don't care
	00	1									
	01 or 10 or 11	1									

Table 329. RTC pin PC13 configuration(1) (continued)

PC13 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E & TAMP2AM	TAMP3E & TAMP3AM & OUT3_RMP	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)	LSCOEN[1:0]
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care	Don't care	Don't care
		00	1									
		01 or 10 or 11	1									
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care	Don't care	Don't care
		00	1									
		01 or 10 or 11	1									
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT2 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care	Don't care	Don't care
TAMP_OUT3 output PP		00	0	0	0	Don't care	Don't care	0	1	Don't care	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	0	0	1	0	Don't care
		00	0	1	1							
		Don't care	Don't care	0								

Table 329. RTC pin PC13 configuration(1) (continued)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E & TAMP2AM	TAMP3E & TAMP3AM & OUT3_RMP	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)	LSCOEN[1:0]
RTC_TS and TAMP_IN1 input floating	00	0	0	Don't care	Don't care	Don't care	0	0	1	1	Don't care
	00	0	1	1			0	0			
	Don't care	Don't care	0				0	0			
RTC_TS input floating	00	0	0	Don't care	Don't care	Don't care	0	0	0	1	Don't care
	00	0	1	1			0	0			
	Don't care	Don't care	0				0	0			
Wakeup pin	00	0	0	Don't care	Don't care	Don't care	0	0	0	0	Don't care
	00	0	1	1			0	0			
	Don't care	Don't care	0				0	0			
LSCO LSE clock output	00	0	0	Don't care	Don't care	Don't care	0	0	0	0	01
	00	0	1	1			0	0			
	Don't care	Don't care	0				0	0			
Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	0	0	00 or 10 or 11
	00	0	1	1			0	0			
	Don't care	Don't care	0				0	0			

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

RTC_OUT2, TAMP2_IN, TAMP3_OUT and RTC_LSCO are mapped on the same pin (PI8). PI8 configuration is controlled by the RTC, whatever the PI8 GPIO configuration. The RTC or TAMP functions mapped on PI8 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in *Table 330*.

Table 330. PI8 configuration

PI8 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN &OUT2_RMP	TAMPALRM_TYPE	TAMPALRM_PU	TAMP3E & TAMP3AM & (OUT3_RMP=0)	TAMP2E	LSCOEN[1:0]
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	1	0	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
TAMPALRM output Open-Drain	No pull	01 or 10 or 11	0	Don't care	1	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	1	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	1	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT3 output Push-Pull		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN2 input floating		00	0	0	0	Don't care	Don't care	0	1	Don't care
Wakeup pin		00	0	0	0	Don't care	Don't care	0	0	Don't care
LSCO LSE clock output		00	0	0	0	Don't care	Don't care	0	0	10
Standard GPIO		00	0	0	0	Don't care	Don't care	0	0	00 or 01 or 11

In addition, it is possible to output RTC_OUT2 on PB2 or PI8 pin thanks to OUT2EN bit. This output is not available in V_{BAT} mode. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table *Table 331*.

For PB2, the GPIO should be configured as an alternate function. This is not the case for PI8, controlled directly by the RTC.

Table 331. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PB2 or PI8
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

48.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and non-secure modes, except for the RTC secure mode control register (RTC_SMCR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the DECPROT bit is cleared in the RTC_SMCR register:

- Writing the RTC registers is possible only in secure mode.
- Reading the RTC registers is possible in secure and non-secure mode.

When the DECPROT bit is set, it is still possible to protect some of the registers by clearing dedicated INITDPROT, CALDPROT, TSDPROT, WUTDPROT, ALRADPROT or

ALRBDPROT control bits. If all these bits are also set, all the RTC registers can be written in secure and non-secure mode.

- When INITDPROT is cleared:
  - RTC_TR, RTC_DR, RTC_PRER and RTC_WPR registers, plus INIT in RTC_ICSR, FMT control bits in RTC_CR can be written only in secure mode.
  - These registers and control bits can be read in secure and non-secure mode.
- When CALDPROT is cleared:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR can be written only in secure mode.
  - These registers and control bits can be read in secure and non-secure mode.
- When ALRADPROT is cleared:
  - RTC_ALRMAR, RTC_ALRMASSR registers, plus ALRAE, ALRAIE in the RTC_CR, CALRAF in the RTC_SCR can be written only in secure mode.
- When ALRBDPROT is cleared:
  - RTC_ALRMBR, RTC_ALRMBSSR registers, plus ALRBE and ALRBIE in the RTC_CR, CALRBF in the RTC_SCR can be written only in secure mode.
- When WUTDPROT is cleared:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR can be written only in secure mode.
- When TSDPROT is cleared:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR can be written only in secure mode.

Accessing a secure-protected register in non-secure mode is done in SILENT mode: the protected bits are not written and are read as 0 without notification.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

48.3.5 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

The RTC clock source (RTCCLK) is selected through the clock controller among the LSE clock, the LSI oscillator clock, and the HSE clock. For more information on the RTC clock source configuration, refer to *Section 10: Reset and clock control (RCC)*.

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 511: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subseconds downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wakeup auto-reload timer. To obtain short timeout periods, the 16-bit wakeup auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 48.3.8: Periodic auto-wakeup* for details).

48.3.6 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 48.6.11: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

48.3.7 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

48.3.8 Periodic auto-wakeup

The periodic wakeup flag is generated by a 16-bit programmable auto-reload down-counter. The wakeup timer range can be extended to 17 bits.

The wakeup function is enabled through the WUTE bit in the RTC_CR register.

The wakeup timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this allows to configure the wakeup interrupt period from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock)

  When ck_spre frequency is 1 Hz, this allows to achieve a wakeup time from 1 s to around 36 hours with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wakeup timer on page 2320*), the timer starts counting down. When the wakeup function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wakeup counter is automatically reloaded with its reload value (RTC_WUTR register value).

The WUTF flag must then be cleared by software.

When the periodic wakeup interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wakeup flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop and Standby) have no influence on the wakeup timer.

48.3.9 RTC initialization and configuration

RTC register access

The RTC registers are 32-bit registers. The APB interface introduces two wait states in RTC register accesses except on read accesses to calendar shadow registers when BYPSHAD = 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected.

Writing to the protected RTC registers is enabled by writing a key into the Write Protection register, RTC_WPR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded and the counting restarts after 4 RTCCLK clock cycles.

When the initialization sequence is complete, the calendar starts counting.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Caution: The active tampers must be disabled before entering the RTC initialization mode (INIT = 1 in the RTC_ICSR). Refer to *Section : Active tamper detection* for more details.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wakeup timer

The following sequence is required to configure or change the wakeup timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wakeup timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wakeup auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode. It takes around 2 RTCCLK clock cycles (due to clock synchronization).
3. Program the wakeup auto-reload value WUT[15:0], and the wakeup clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wakeup timer restarts down-counting.The WUTWF bit is cleared up to 2 RTCCLK clocks cycles after WUTE is cleared, due to clock synchronization.

48.3.10 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB1 clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB1 clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB1 clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycles. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wakeup and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 2319*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 48.3.12: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

48.3.11 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wakeup timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR and RTC_ALRMBSSR/RTC_ALRMBR), and the configuration register (RTC_CFGR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

48.3.12 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the sub-second field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

RTC_SSR contains the value of the synchronous prescaler counter. This allows one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]. The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock by up to a second with a resolution of 1 / (PREDIV_S + 1) seconds. The shift operation consists of adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this will delay the clock. If at the same time the ADD1S bit is set, this results in adding one second and at the same time subtracting a fraction of second, so this will advance the clock.

Caution: Before initiating a shift operation, the user must check that SS[15] = 0 in order to ensure that no overflow will occur.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

48.3.13 RTC reference clock detection

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

48.3.14 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual RTCCLK pulses). These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

The smooth digital calibration is performed during a cycle of about 2^{20} RTCCLK pulses, or 32 seconds when the input frequency is 32768 Hz. This cycle is maintained by a 20-bit counter, cal_cnt[19:0], clocked by RTCCLK.

The smooth calibration register (RTC_CALR) specifies the number of RTCCLK clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of RTCCLK pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM allows the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra RTCCLK pulse every 2^{11} RTCCLK cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 RTCCLK cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

To perform a calibration with PREDIV_A less than 3, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 RTCCLK clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

48.3.15 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a tamper event is detected on the TAMP_INx pinx.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set 2 ck_apre cycles after the timestamp event occurs due to synchronization process.

There is no delay in the setting of TSOVF. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

Optionally, a tamper event can cause a timestamp to be recorded. See the description of the TAMPTS control bit in the RTC control register (RTC_CR).

48.3.16 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is fRTCCLK/(256 * (PREDIV_A+1)). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When the CALIB output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function.

When COSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler.

When COSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

48.3.17 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wakeup flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

When TAMPALRM output is selected, the RTC_OUT1 pin is automatically configured but the RTC_OUT2 pin must be set as alternate function.

48.4 RTC low-power modes

Table 332. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 333. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby modes	Functional in Standby mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	No	on PB2: No
			on PI8: Yes
LSCO	Yes	Yes	Yes

48.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The interrupt output or the secure interrupt output is also activated.

Table 334. Non-secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
RTC_WKUP_ALARM	Alarm A	ALRAF	ALRAIE and (ALRADPROT=1 and DECPROT=1)	write 1 in CALRAF	Yes	Yes(3)
RTC_WKUP_ALARM	Alarm B	ALRBF	ALRBIE and (ALRBDPROT=1 and DECPROT=1)	write 1 in CALRBF	Yes	Yes(3)

Table 334. Non-secure interrupt requests (continued)

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
RTC_TS	Timestamp	TSF	TSIE and (TSDPROT=1 and DECPROT=1)	write 1 in CTSF	Yes	Yes(3)
RTC_WKUP_ALARM	Wakeup timer	WUTF	WUTIE and (WUTDPROT=1 and DECPROT=1)	write 1 in CWUTF	Yes	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. Wakeup from Stop and Standby modes is possible only when the RTC clock source is LSE or LSI.

Table 335. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
RTC_WKUP_ALARM_S	Alarm A	ALRAF	ALRAIE and (ALRADPROT=0 or DECPROT=0)	write 1 in CALRAF	Yes	Yes(3)
RTC_WKUP_ALARM_S	Alarm B	ALRBF	ALRBIE and (ALRBDPROT=0 or DECPROT=0)	write 1 in CALRBF	Yes	Yes(3)
RTC_TS_S	Timestamp	TSF	TSIE and (TSDPROT=0 or DECPROT=0)	write 1 in CTSF	Yes	Yes(3)
RTC_WKUP_ALARM_S	Wakeup timer	WUTF	WUTIE and (WUTDPROT=0 or DECPROT=0)	write 1 in CWUTF	Yes	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. Wakeup from Stop and Standby modes is possible only when the RTC clock source is LSE or LSI.

48.6 RTC registers

Refer to *Section 1.2 on page 115* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

48.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 2319* and *Reading the calendar on page 2320*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be write-protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

48.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 2319* and *Reading the calendar on page 2320*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be write-protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

48.6.3 RTC sub second register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SS[15:0]**: Sub second value

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

Note: SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

48.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset value: 0bxxxx xxxx xxxx xxxx xxxx xxxx 000x xxxx (not affected, except INIT, INITF, and RSF bits which are cleared to 0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
								rw	r	rc_w0	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER). Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wakeup timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wakeup timer configuration update not allowed except in initialization mode

1: Wakeup timer configuration update allowed

Bit 1 **ALRBWF**: Alarm B write flag

This bit is set by hardware when alarm B values can be changed, after the ALRBE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm B update not allowed

1: Alarm B update allowed

Bit 0 **ALRAWF**: Alarm A write flag

This bit is set by hardware when alarm A values can be changed, after the ALRAE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Alarm A update not allowed

1: Alarm A update allowed

48.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 2319*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be write-protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor
This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor
This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

48.6.6 RTC wakeup timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WUT[15:0]**: Wakeup auto-reload value bits
When the wakeup timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.
When WUCKSEL[2] = 1, the wakeup timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.
The first assertion of WUTF occurs between WUT and (WUT + 1) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

48.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 2319.

This register can be globally protected, or each bit of this register can be individually protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2 EN	TAMP ALRM_ TYPE	TAMP ALRM_ PU	Res.	Res.	TAMP OE	TAMP TS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRB IE	ALRA IE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYP SHAD	REFCK ON	TS EDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

Setting this bit allows to remap the RTC outputs on RTC_OUT2 as follows:

OUT2EN = 0: RTC output 2 disable

If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1

OUT2EN = 1: RTC output 2 enable

If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2

If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2

If (OSEL≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output

1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output

1: A pull-up is applied on TAMPALRM output

Bits 28:27 Reserved, must be kept at reset value.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM

0: The tamper flag is not routed on TAMPALRM

1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

0: Tamper detection event does not cause a RTC timestamp to be saved

1: Save RTC timestamp on tamper detection event

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set after the tamper flags, therefore if TAMPTS and TSIE are set, it is recommended to disable the tamper interrupts in order to avoid servicing 2 interrupts.

Bit 24 **ITSE**: timestamp on internal event enable

0: internal event timestamp disabled

1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable
This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wakeup output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 48.3.16: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: *S*ubtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wakeup timer interrupt enable
0: Wakeup timer interrupt disabled
1: Wakeup timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wakeup timer enable
0: Wakeup timer disabled
1: Wakeup timer enabled
Note: When the wakeup timer is disabled, wait for WUTWF=1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB1 clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wakeup clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected
11x: ck_spre (usually 1 Hz) clock is selected and 2^{16} is added to the WUT counter value

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wakeup unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it could mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

48.6.8 RTC secure mode control register (RTC_SMCR)

This register can be written only when the APB access is secure.

Address offset: 0x20

Backup domain reset value: 0x0000 E00F

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DEC PROT	INIT DPROT	CAL DPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS DPROT	WUT DPROT	ALRB DPROT	ALRA DPROT
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DECPROT**: RTC global protection

0: All RTC registers can be written only when the APB access is secure.

1: All RTC registers can be written when the APB access is secure or non-secure, except the registers protected by other secure protection bits.

Bit 14 **INITDPROT**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

1: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or non-secure.

Bit 13 **CALDPROT**: Shift register, daylight saving, calibration and reference clock protection

0: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

1: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or non-secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSDPROT**: Timestamp protection

0: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

1: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or non-secure.

Bit 2 **WUTDPROT**: Wakeup timer protection

0: RTC wakeup timer configuration and interrupt clear can be written only when the APB access is secure.

1: RTC wakeup timer configuration and interrupt clear can be written when the APB access is secure or non-secure.

Bit 1 **ALRBDPROT**: Alarm B protection

0: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

1: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or non-secure.

Bit 0 **ALRADPROT**: Alarm A protection

0: RTC alarm A configuration and interrupt clear can be written only when the APB access is secure.

1: RTC alarm A configuration and interrupt clear can be written when the APB access is secure or non-secure.

Note: Refer to Section 48.3.4: RTC secure protection modes for details on the read protection.

48.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

48.6.10 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be write-protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCLK pulses are added.

1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows: (512 × CALP) - CALM.

Refer to *Section 48.3.14: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 48.3.14: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 48.3.14: RTC smooth digital calibration.

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 48.3.14: RTC smooth digital calibration on page 2323*.

48.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

48.6.12 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
- 0: AM or 24-hour format
- 1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

48.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

48.6.14 RTC timestamp sub second register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 SS[15:0]: Sub second value
SS[15:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

48.6.15 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

48.6.16 RTC alarm A sub second register (RTC_ALRMASSR)

This register can be written only when ALRAWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit

0: No comparison on sub seconds for alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[14:1] are don't care in alarm A comparison. Only SS[0] is compared.
2: SS[14:2] are don't care in alarm A comparison. Only SS[1:0] are compared.
3: SS[14:3] are don't care in alarm A comparison. Only SS[2:0] are compared.
...
12:SS[14:12] are don't care in alarm A comparison. SS[11:0] are compared.
13:SS[14:13] are don't care in alarm A comparison. SS[12:0] are compared.
14:SS[14] is don't care in alarm A comparison. SS[13:0] are compared.
15:All 15 SS bits are compared and must match to activate alarm.

The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Note: The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

48.6.17 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBWF is set to 1 in RTC_ICSR, or in initialization mode.

This register is write protected. The write access procedure is described in *RTC register write protection on page 2319*.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask

0: Alarm B set if the date and day match
1: Date and day don't care in alarm B comparison

Bit 30 **WDSEL**: Week day selection

0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

48.6.18 RTC alarm B sub second register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register is write protected.The write access procedure is described in *Section : RTC register write protection*.

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **MASKSS[3:0]**: Mask the most-significant bits starting at this bit
0x0: No comparison on sub seconds for alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
0x1: SS[14:1] are don't care in alarm B comparison. Only SS[0] is compared.
0x2: SS[14:2] are don't care in alarm B comparison. Only SS[1:0] are compared.
0x3: SS[14:3] are don't care in alarm B comparison. Only SS[2:0] are compared.
...
0xC: SS[14:12] are don't care in alarm B comparison. SS[11:0] are compared.
0xD: SS[14:13] are don't care in alarm B comparison. SS[12:0] are compared.
0xE: SS[14] is don't care in alarm B comparison. SS[13:0] are compared.
0xF: All 15 SS bits are compared and must match to activate alarm.
The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Sub seconds value
This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

48.6.19 RTC status register (RTC_SR)

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ITSF**: Internal timestamp flag
This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag
This flag is set by hardware when a timestamp event occurs while TSF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag
This flag is set by hardware when a timestamp event occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTF**: Wakeup timer flag

This flag is set by hardware when the wakeup auto-reload counter reaches 0.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

48.6.20 RTC non-secure masked interrupt status register (RTC_MISR)

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ITSMF**: Internal timestamp non-secure masked flag

This flag is set by hardware when a timestamp on the internal event occurs and timestamp non-secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow non-secure masked flag

This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp non-secure masked flag

This flag is set by hardware when a timestamp non-secure interrupt occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wakeup timer non-secure masked flag

This flag is set by hardware when the wakeup timer non-secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B non-secure masked flag

This flag is set by hardware when the alarm B non-secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag

This flag is set by hardware when the alarm A non-secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

48.6.21 RTC secure masked interrupt status register (RTC_SMISR)

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x58

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITS MF	TSOV MF	TS MF	WUT MF	ALRB MF	ALRA MF
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **ITSMF**: Internal timestamp interrupt secure masked flag

This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag

This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag

This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wakeup timer interrupt secure masked flag
This flag is set by hardware when the wakeup timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

48.6.22 RTC status clear register (RTC_SCR)

This register can be protected against non-secure access. Refer to *Section 48.3.4: RTC secure protection modes*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
										w	w	w	w	w	w

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wakeup timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

48.6.23 RTC configuration register (RTC_CFGR)

Address offset: 0x60

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSCOEN[1:0]		OUT2_ RMP
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:1 **LSCOEN[1:0]**: LSCO LSE clock output enable
- 00: no effect
- 01: LSE is output on PC13
- 10: LSE is output on PI8
- 11: no effect

Note: Note: LSCO function is output on PC13 or PI8 only if no other RTC output is enabled on these I/Os.

Bit 0 **OUT2_RMP**: RTC_OUT2 mapping
- 0: RTC_OUT2 is mapped on PB2
- 1: RTC_OUT2 is mapped on PI8

48.6.24 RTC hardware configuration register (RTC_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0103 1111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TRUST_ZONE[3:0]				OPTIONREG_OUT[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TIMESTAMP[3:0]				SMOOTH_CALIB[3:0]				WAKEUP[3:0]				ALARMB[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TRUST_ZONE[3:0]**
0x0: Trust zone not implemented
0x1: Trust zone implemented

Bits 23:16 **OPTIONREG_OUT[7:0]**
0: Configuration register not implemented
1: Configuration register bit 0 implemented
2: Configuration register bits 1:0 implemented
...
32: Configuration register bits 31:0 implemented

Bits 15:12 **TIMESTAMP[3:0]**
0x0: Timestamp not implemented
0x1: Timestamp implemented

Bits 11:8 **SMOOTH_CALIB[3:0]**
0x0: Smooth calibration not implemented
0x1: Smooth calibration implemented

Bits 7:4 **WAKEUP[3:0]**
0x0: Wakeup timer not implemented
0x1: Wakeup timer implemented

Bits 3:0 **ALARMB[3:0]**
0x0: Alarm B not implemented
0x1: Alarm B implemented

48.6.25 RTC version register (RTC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 001X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

48.6.26 RTC identification register (RTC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 0033

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier

48.6.27 RTC size identification register (RTC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier

48.6.28 RTC register map

Table 336. RTC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INIT	INITF	RSF	INITS	SHPF	WUT WF	ALRBWF	ALRAWF
	Reset value																0									0	0	0	0	0	1	1	1
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	Res.	Res.	TAMPOE	TAMPTS	ITSE	COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	Res.	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
	Reset value	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x20	RTC_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DECPROT	INITDPROT	CALDPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSDPROT	WUTDPROT	ALRBDPROT	ALRADPROT
	Reset value																	1	1	1										1	1	1	1
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	Res.	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0					0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0

Table 336. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[1:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	Res.	Res.	Res.	Res.	MASKSS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																											0	0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																											0	0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																											0	0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																											0	0	0	0	0	0
0x60	RTC_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSCOEN[1:0]		OUT2_RMP
	Reset value																														0	0	0
0x3F0	RTC_ HWCFGR	Res.	Res.	Res.	Res.	TRUST_ ZONE[3:0]				OPTION_REG_OUT[7:0]								TIMESTAMP [3:0]				SMOOTH_ CALIB[3:0]				WAKEUP [3:0]				ALARMB [3:0]			
	Reset value					0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
0x3F4	RTC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	x	x	x	x
0x3F8	RTC_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1

Table 336. RTC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3FC	**RTC_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

49 Tamper and backup registers (TAMP)

49.1 Introduction

32 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. They can be used to store sensitive data as their content is protected by an tamper detection circuit. 3 tamper pins and 6 internal tampers are available for anti-tamper detection. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

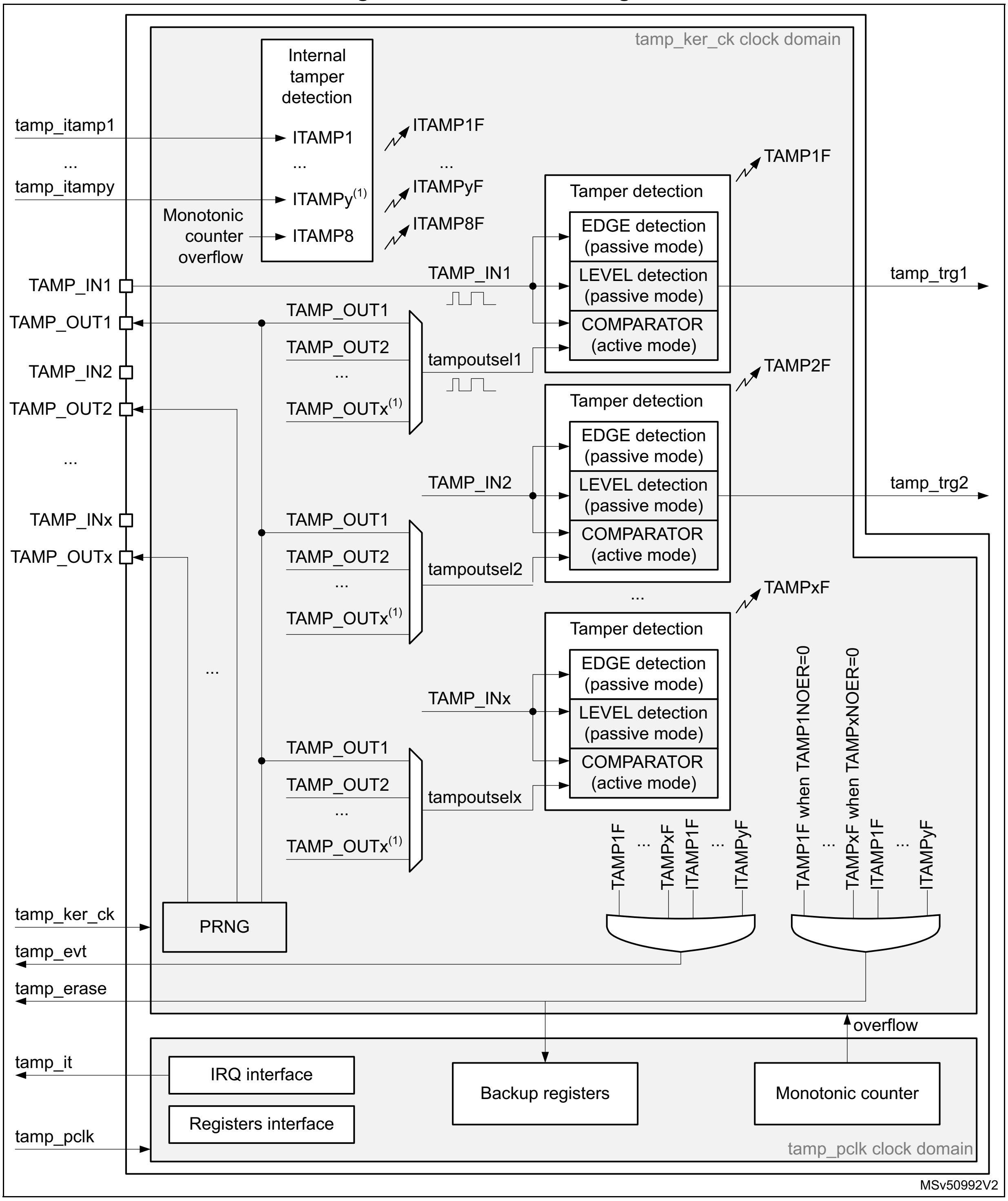
49.2 TAMP main features

- 32 backup registers:
  - the backup registers (TAMP_BKPxR) are implemented in the RTC domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- 3 external tamper detection events.
  - Each external event can be configured to be active or passive.
  - External passive tampers with configurable filter and internal pull-up.
- 6 internal tamper events.
- Any tamper detection can generate a RTC timestamp event.
- Any tamper detection can erase the backup registers and backup SRAM.
- TrustZone support:
  - Tamper secure or non-secure configuration.
  - Backup registers configuration in 3 configurable-size areas:
    1 read/write secure area.
    1 write secure/read non-secure area.
    1 read/write non-secure area.
- Monotonic counter.

49.3 TAMP functional description

49.3.1 TAMP block diagram

Figure 512. TAMP block diagram



1. The number of external and internal tampers depends on products.

49.3.2 TAMP pins and internal signals

Table 337. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin

Table 338. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_evt	Output	Tamper event detection (internal or external) The tamp_evt is used to generate a RTC timestamp event
tamp_erase	Output	Device secrets erase request following tamper event detection (internal or external)
tamp_it	Output	TAMP interrupt (refer to *Section 49.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or V_{BAT} when the selected clock is not LSE (refer to *Section 49.4: TAMP low-power modes* for more details.

Table 339. TAMP interconnection

Signal name	Source/Destination
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_erase	The tamp_erase signal is used to erase the device secrets listed hereafter: backup registers and backup SRAM
tamp_itamp1	RTC voltage domain monitoring
tamp_itamp2	Temperature monitoring
tamp_itamp3	LSE monitoring
tamp_itamp4	HSE monitoring
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp8(1)	Monotonic counter overflow

1. This signal is generated in the TAMP peripheral.

49.3.3 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

49.3.4 TAMP secure protection modes

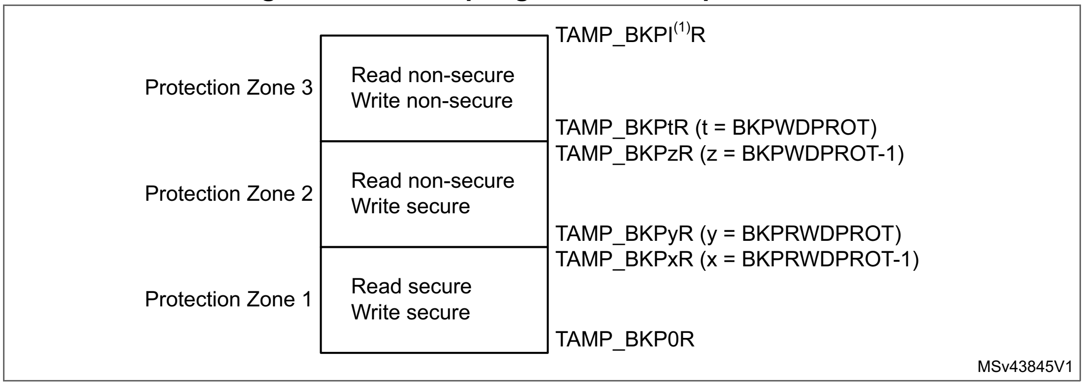
By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and non-secure modes, except for the TAMP secure mode control register (TAMP_SMCR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

When the TAMPDPROT bit is cleared in the TAMP_SMCR register:

- Writing the TAMP registers is possible only in secure mode, except for the backup registers which have their own protection setting.
- Reading the registers is always possible in secure and non-secure modes, except for the PRNG field of the TAMP_ATOR register which can be read only is secure mode.

The backup registers protection is configured thanks to BKPRWDPROT[7:0] and BKPWDPROT[7:0] (refer to *Figure 513* below):

Figure 513. Backup registers secure protections



1. I= last backup register index

Accessing a secure-protected register in non-secure mode is done in SILENT mode: read access in non-secure mode to protected TAMP registers returns 0, and write access in non-secure mode to protected TAMP registers is ignored, without notification.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

49.3.5 Tamper detection

The tamper detection can be configured for the following purposes:

- erase the backup registers and the SRAMs listed in *Table 339: TAMP interconnection* (default configuration)
- generate an interrupt, capable to wakeup from Stop and Standby mode
- generate a hardware trigger for the low-power timers

TAMP backup registers

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers are reset when a tamper detection event occurs except if the TAMPxNOER bit is set, or if the TAMPxMSK is set in the TAMP_CR2 register.

Tamper detection initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

When TAMPxMSK is cleared:

The TAMPxF flag is asserted after the tamper event on the pin, with the latency provided below:

- 3 ck_apre cycles when TAMPFLT differs from 0x0 (level detection with filtering)
- 3 ck_apre cycles when TAMPTS = 1 (timestamp on tamper event)
- No latency when TAMPFLT = 0x0 (edge detection) and TAMPTS = 0

A new tamper occurring on the same pin during this period and as long as TAMPxF is set cannot be detected.

When TAMPxMSK is set:

A new tamper occurring on the same pin cannot be detected during the latency described above and 2.5 ck_rtc additional cycles.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit in cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration allows to trig automatically the low-power timers in Stop mode, without requiring the system wakeup to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in the *Level detection with filtering on tamper inputs (passive mode)* mode (TAMPFLT ≠ 00 and active mode is not selected).

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any tamper event causes a timestamp to occur. In this case, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs. The affected tamper flag register TAMPxF is set in the TAMP_SR at the same time that TSF or TSOVF is set in the RTC_SR.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge/high level or a falling edge/low level is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When using the edge detection, it is recommended to check by software the tamper pin level just after enabling the tamper detection (by reading the GPIO registers), and before writing sensitive values in the backup registers, to ensure that an active edge did not occur before enabling the tamper event detection.
When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCR register.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} x CK_ATPRE = 2^{ATPER} x $2^{ATCKSEL}$ RTCCLK), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.
PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on

TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 340. Minimum ATPER value

Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, taking into account feedback delay. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 00 and ATOSEL2 = 00.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in RTC_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):

  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the RTC APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in RTC_ATOR. Backup registers are then protected by active tamper.

Caution: The active tampers must be disabled before entering the RTC initialization mode (INIT = 1 in the RTC_ICSR). Refer to *Section : Calendar initialization and configuration* for more details.

49.4 TAMP low-power modes

Table 341. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode.

49.5 TAMP interrupts

The interrupt channel is set in the interrupt status register or in the secure interrupt status register depending on its secure mode (TAMPDPROT) configuration. The interrupt output or the secure interrupt output is also activated.

Table 342. Non-secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby modes
TAMP	Tamper x(3)	TAMPxF	TAMPxIE and TAMPDPROT=1	Write 1 in CTAMPxF	Yes	Yes(4)
	Internal tamper y(3)	ITAMPyF	TAMPyIE and TAMPDPROT=1	Write 1 in CITAMPxF	Yes	Yes(4)

1. The event flags are in the TAMP_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the TAMP_MISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, or in case of active tamper mode, wakeup from Stop and Standby modes is possible only when the TAMP clock source is LSE or LSI.

Table 343. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby modes
TAMP_S	Tamper x(3)	TAMPxF	TAMPxIE and TAMPDPROT=0	Write 1 in CTAMPxF	Yes	Yes(4)
	Internal tamper y(3)	ITAMPyF	TAMPyIE and TAMPDPROT=0	Write 1 in CITAMPxF	Yes	Yes(4)

1. The event flags are in the TAMP_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the TAMP_SMISR register.
3. The number of tampers and internal tampers events depend on products.
4. In case of level detection with filtering passive tamper mode, or in case of active tamper mode, wakeup from Stop and Standby modes is possible only when the TAMP clock source is LSE or LSI.

49.6 TAMP registers

Refer to *Section 1.2 on page 115* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

49.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP*

secure protection modes.

Address offset: 0x00

Backup domain reset value: 0xFFFF 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8E	Res.	Res.	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
								rw			rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E
													rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8E**: Internal tamper 8 enable: monotonic counter overflow

0: Internal tamper 8 disabled.

1: Internal tamper 8 enabled: a tamper is generated when the TAMP_CNT overflows.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5E**: Internal tamper 5 enable: RTC calendar overflow

0: Internal tamper 5 disabled.

1: Internal tamper 5 enabled: a tamper is generated when the RTC calendar reaches its maximum value, on the 31^{st} of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

Bit 19 **ITAMP4E**: Internal tamper 4 enable: HSE monitoring

0: Internal tamper 4 disabled.

1: Internal tamper 4 enabled. a tamper is generated when the HSE frequency is below or above thresholds.

Bit 18 **ITAMP3E**: Internal tamper 3 enable: LSE monitoring

0: Internal tamper 3 disabled.

1: Internal tamper 3 enabled: a tamper is generated when the LSE frequency is below or above thresholds.

Bit 17 **ITAMP2E**: Internal tamper 2 enable: Temperature monitoring

0: Internal tamper 2 disabled.

1: Internal tamper 2 enabled: a tamper is generated when the temperature is below or above thresholds.

Bit 16 **ITAMP1E**: Internal tamper 1 enable: RTC power domain supply monitoring

0: Internal tamper 1 disabled.

1: Internal tamper 1 enabled: a tamper is generated when the RTC power domain supply is below or above thresholds.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
- 0: Tamper detection on TAMP_IN3 is disabled.
- 1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
- 0: Tamper detection on TAMP_IN2 is disabled.
- 1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
- 0: Tamper detection on TAMP_IN1 is disabled.
- 1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

49.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP secure protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	Res.	Res.	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
					rw	rw	rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3 NOER	TAMP2 NOER	TAMP1 NOER
													rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 Tamper 3 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 Tamper 3 input rising edge and high level triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 Tamper 3 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 Tamper 3 input falling edge and low level triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 Tamper 2 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 Tamper 2 input rising edge and high level triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 Tamper 2 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 Tamper 2 input falling edge and low level triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input (active mode disabled)

0: If TAMPFLT ≠ 00 Tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input rising edge and high level triggers a tamper detection event.

1: If TAMPFLT ≠ 00 Tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 Tamper 1 input falling edge and low level triggers a tamper detection event.

Bit 23 Reserved, must be kept at reset value.

Bits 22:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3NOER**: Tamper 3 no erase

0: Tamper 3 event erases the backup registers.

1: Tamper 3 event does not erase the backup registers(1).

Bit 1 **TAMP2NOER**: Tamper 2 no erase

0: Tamper 2 event erases the backup registers.

1: Tamper 2 event does not erase the backup registers(1).

Bit 0 **TAMP1NOER**: Tamper 1 no erase

0: Tamper 1 event erases the backup registers.

1: Tamper 1 event does not erase the backup registers(1).

1. and the device secrets erased by tamp_erase signal (refer to *Table 339: TAMP interconnection*).

49.6.3 TAMP filter control register (TAMP_FLTCR)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP*

secure protection modes.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.

0: Precharge TAMP_INx pins before sampling (enable internal pull-up)

1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.

0x0: 1 RTCCLK cycle

0x1: 2 RTCCLK cycles

0x2: 4 RTCCLK cycles

0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.

0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).

0x1: Tamper event is activated after 2 consecutive samples at the active level.

0x2: Tamper event is activated after 4 consecutive samples at the active level.

0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.

0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)

0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)

0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)

0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)

0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)

0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)

0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)

0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

49.6.4 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP secure protection modes*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	Res.	ATCKSEL[2:0]		
rw	rw				rw	rw	rw						rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	Res.	Res.	Res.	Res.	TAMP3 AM	TAMP2 AM	TAMP1 AM
		rw	rw	rw	rw	rw	rw						rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable
0: Active tamper filtering disable
1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing
0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi
1: Each active tamper input TAMP_INi is compared with TAMPOUTSELx as defined below, with TAMPOUTSELx defined by ATOSELx bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period
The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 340: Minimum ATPER value*.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **ATCKSEL[2:0]**: Active tamper RTC asynchronous prescaler clock selection
These bits selects the RTC asynchronous prescaler stage output.The selected clock is CK_ATPRE.
$f_{CK\_ATPRE} = f_{RTCCLK} / 2^{ATCKSEL}$ when (PREDIV_A+1) = 128.
000: RTCCLK is selected
001: RTCCLK/2 is selected when (PREDIV_A+1) = 128 (actually selects 1st flip flop output)
010: RTCCLK/4 is selected when (PREDIV_A+1) = 128 (actually selects 2nd flip flop output)
...
111: RTCCLK/128 is selected when (PREDIV_A+1) = 128 (actually selects 7th flip flop output)
Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 ck_atpre cycles after all the active tampers are disable.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection
00: TAMPOUTSEL3 = TAMP_OUT1
01: TAMPOUTSEL3 = TAMP_OUT2
10: TAMPOUTSEL3 = TAMP_OUT3
11:
The selected output must be available in the package pinout

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection
00: TAMPOUTSEL2 = TAMP_OUT1
01: TAMPOUTSEL2 = TAMP_OUT2
10: TAMPOUTSEL2 = TAMP_OUT3
11:
The selected output must be available in the package pinout

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection
00: TAMPOUTSEL1 = TAMP_OUT1
01: TAMPOUTSEL1 = TAMP_OUT2
10: TAMPOUTSEL1 = TAMP_OUT3
11:
The selected output must be available in the package pinout

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

49.6.5 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP*

secure protection modes.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEED[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

49.6.6 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP secure protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
r	r							r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is left unchanged when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP secure protection modes*.

49.6.7 TAMP secure mode register (TAMP_SMCR)

This register can be written only when the APB access is secure.

Address offset: 0x20

Backup domain reset value: 0x8000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPDPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWDPROT[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWDPROT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPDPROT**: Tamper protection (excluding backup registers)

0: Tamper configuration and interrupt can be written only when the APB access is secure.

1: Tamper configuration and interrupt can be written when the APB access is secure or non-secure.

Note: Refer to Section 49.3.4: TAMP secure protection modes for details on the read protection.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWDPROT[7:0]**: Backup registers write protection offset

Backup registers from TAMP_BKPyR (y = BKPRWDPROT, from 0 to 32) to TAMP_BKPzR (z = BKPWDPROT-1, from 0 to 32, BKPWDPROT ≥ BKPRWDPROT) can be written only when the APB is in secure mode. They can be read in secure or non-secure mode. This zone is the protection zone 2.

Backup registers from TAMP_BKPtR (t = BKPWDPROT, from 0 to 31) can be read or written when the APB is in secure or in non-secure mode. This zone is the protection zone 3.

If BKPWDPROT = 0 or if BKPWDPROT ≤ BKPRWDPROT: none of the backup registers have a secure write access. In these configurations the behavior is equivalent to BKPWDPROT = BKPRWDPROT.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWDPROT[7:0]**: Backup registers read/write protection offset

Backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWDPROT-1, from 0 to 32) can be read and written only when the APB is in secure mode. This is the protection zone 1.

If BKPRWSM = 0 none of the backup registers have a secure read/write access.

49.6.8 TAMP interrupt enable register (TAMP_IER)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP*

secure protection modes.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8IE	Res.	Res.	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
								rw			rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
													rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable: monotonic counter overflow
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable: RTC calendar overflow
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable: HSE monitoring
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable: LSE monitoring
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable: Temperature monitoring
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable: RTC power domain supply monitoring
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

49.6.9 TAMP status register (TAMP_SR)

This register can be protected against non-secure access. Refer to *Section 49.3.4: TAMP secure protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8F	Res.	Res.	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
								r			r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
													r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8F**: Monotonic counter overflow tamper flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5F**: RTC calendar overflow tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: HSE monitoring tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: LSE monitoring tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Temperature monitoring tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: RTC power domain voltage monitoring tamper detection flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

49.6.10 TAMP non-secure masked interrupt status register (TAMP_MISR)

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	Res.	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
								r			r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
													r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8MF**: Monotonic counter overflow non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 non-secure interrupt is raised.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5MF**: RTC calendar overflow tamper non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 non-secure interrupt is raised.

Bit 19 **ITAMP4MF**: HSE monitoring tamper non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 4 non-secure interrupt is raised.

Bit 18 **ITAMP3MF**: LSE monitoring tamper non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 non-secure interrupt is raised.

Bit 17 **ITAMP2MF**: Temperature monitoring tamper non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 non-secure interrupt is raised.

Bit 16 **ITAMP1MF**: RTC power domain voltage monitoring tamper non-secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 non-secure interrupt is raised.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3MF**: TAMP3 non-secure interrupt masked flag
This flag is set by hardware when the tamper 3 non-secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 non-secure interrupt masked flag
This flag is set by hardware when the tamper 2 non-secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 non-secure interrupt masked flag
This flag is set by hardware when the tamper 1 non-secure interrupt is raised.

49.6.11 TAMP secure masked interrupt status register (TAMP_SMISR)

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8 MF	Res.	Res.	ITAMP5 MF	ITAMP4 MF	ITAMP3 MF	ITAMP2 MF	ITAMP1 MF
								r			r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP 3MF	TAMP 2MF	TAMP 1MF
													r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ITAMP8MF**: Monotonic counter overflow secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ITAMP5MF**: RTC calendar overflow tamper secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 **ITAMP4MF**: HSE monitoring tamper secure interrupt masked flag
This flag is set by hardware when the internal tamper 4 secure interrupt is raised.

Bit 18 **ITAMP3MF**: LSE monitoring tamper secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 **ITAMP2MF**: Temperature monitoring tamper secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 secure interrupt is raised.

Bit 16 **ITAMP1MF**: RTC power domain voltage monitoring tamper secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

49.6.12 TAMP status clear register (TAMP_SCR)

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP8F	Res.	Res.	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F
								w			w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP3F	CTAMP2F	CTAMP1F
													w	w	w

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

49.6.13 TAMP monotonic counter register (TAMP_COUNTR)

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COUNT[31:0]**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

49.6.14 TAMP configuration register (TAMP_CFGR)

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT3_RMP
															rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **OUT3_RMP**: TAMP_OUT3 mapping

0x0: TAMP_OUT3 is mapped on PI8

0x1: TAMP_OUT3 is mapped on PC13

49.6.15 TAMP backup x register (TAMP_BKPxR)

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 **BKP[31:0]**
The application can write or read data to and from these registers.
They are powered-on by V_{BAT} when V_{DD} is switched off, so that they are not reset by System reset, and their contents remain valid when the device operates in low-power mode.
In the default configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set.

49.6.16 TAMP hardware configuration register 2 (TAMP_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TRUST_ZONE[3:0]				OPTIONREG_OUT[7:0]							
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TRUST_ZONE[3:0]**
0x0: Trust_zone not supported
0x1: Trust_zone supported

Bits 7:0 **OPTIONREG_OUT[7:0]**
0: Configuration register not implemented
1: Configuration register bit 0 implemented
2: Configuration register bits 1:0 implemented
...
32: Configuration register bits 31:0 implemented

49.6.17 TAMP hardware configuration register 1 (TAMP_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x009D 1320

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT_TAMPER[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE_TAMPER[3:0]				TAMPER[3:0]				BACKUP_REGS[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **INT_TAMPER[15:0]**

INT_TAMPER[i] = 0: internal tamper i is not implemented (i from 0 to 15)
INT_TAMPER[i] = 1: internal tamper i is implemented (i from 0 to 15)

Bits 15:12 **ACTIVE_TAMPER[3:0]**

0x0: Active tamper not supported
0x1: Active tamper supported for all external tamper events

Bits 11:8 **TAMPER[3:0]**

0x0: 16 external tamper events (input pins) supported
0x1: 1 external tamper event (input pin) supported
0x2: 2 external tamper events (input pins) supported
...
0xF: 15 external tamper events (input pins) supported

Bits 7:0 **BACKUP_REGS[7:0]**

0: No backup register
1: 1 Backup register implemented
2: 2 Backup registers implemented
...
128: 128 backup registers implemented

49.6.18 TAMP version register (TAMP_VERR)

Address offset: 0x3F4

Reset value: 0x0000 001X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

49.6.19 TAMP identification register (TAMP_IPIDR)

Address offset: 0x3F8

Reset value: 0x0012 1033

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier.

49.6.20 TAMP size identification register (TAMP_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier.

49.6.21 TAMP register map

Table 344. TAMP register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TAMP_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8E	Res.	Res.	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3E	TAMP2E	TAMP1E
	Reset value									1			1	1	1	1	1														0	0	0
0x04	**TAMP_CR2**	Res.	Res.	Res.	Res.	Res.	TAMP3TRG	TAMP2TRG	TAMP1TRG	Res.	Res.	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3NOER	TAMP2NOER	TAMP1NOER
	Reset value						0	0	0						0	0	0														0	0	0
0x0C	**TAMP_FLTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	**TAMP_ATCR1**	FLTEN	ATOSHARE	Res.	Res.	Res.	AT-PER[2:0]			Res.	Res.	Res.	Res.	Res.	ATCK-SEL[2:0]			Res.		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		Res.	Res.	Res.	Res.	Res.	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0						1	1	1			0	0	0	0	0	0						0	0	0
0x14	**TAMP_ ATSEEDR**	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**TAMP_ATOR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x20	**TAMP_SMCR**	TAMPDPROT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPWDPROT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWDPROT[7:0]							
	Reset value	1								0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x2C	**TAMP_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8IE	Res.	Res.	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value									0			0	0	0	0	0														0	0	0
0x30	**TAMP_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8F	Res.	Res.	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3F	TAMP2F	TAMP1F
	Reset value									0			0	0	0	0	0														0	0	0
0x34	**TAMP_MISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	Res.	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value									0			0	0	0	0	0														0	0	0

Table 344. TAMP register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	**TAMP_SMISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP8MF	Res.	Res.	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value									0			0	0	0	0	0														0	0	0
0x3C	**TAMP_SCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CITAMP8F	Res.	Res.	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value									0			0	0	0	0	0														0	0	0
0x40	**TAMP_ COUNTR**	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	**TAMP_CFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT3_RM
	Reset value																																0
0x100 + 0x04*x, (x = 0 to 31)	**TAMP_BKPxR**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3EC	**TAMP_ HWCFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRUST_ ZONE[3:0]				OPTIONREG_OUT[7:0]							
	Reset value																					0	0	0	1	0	0	0	0	0	0	0	1
0x3F0	**TAMP_ HWCFGR1**	INT_TAMPER[15:0]																ACTIVE_ TAMPER [3:0]				TAMPER [3:0]				BACKUP_REGS [7:0]							
	Reset value	0	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0
0x3F4	**TAMP_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	x	x	x	x
0x3F8	**TAMP_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	1	1
0x3FC	**TAMP_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

50 Inter-integrated circuit interface (I2C)

50.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

50.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[a]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to *Section 50.3: I2C implementation*.

50.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 345. I2C implementation

I2C features(1)	I2C1	I2C2	I2C3	I2C4	I2C5	I2C6
7-bit addressing mode	X	X	X	X	X	X
10-bit addressing mode	X	X	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X	X	X
Independent clock	X	X	X	X	X	X
Wake-up from Stop mode	X	X	X	X	X	X
SMBus/PMBus	X	X	X	X	X	X

1. X = supported.

50.4 I2C functional description

In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

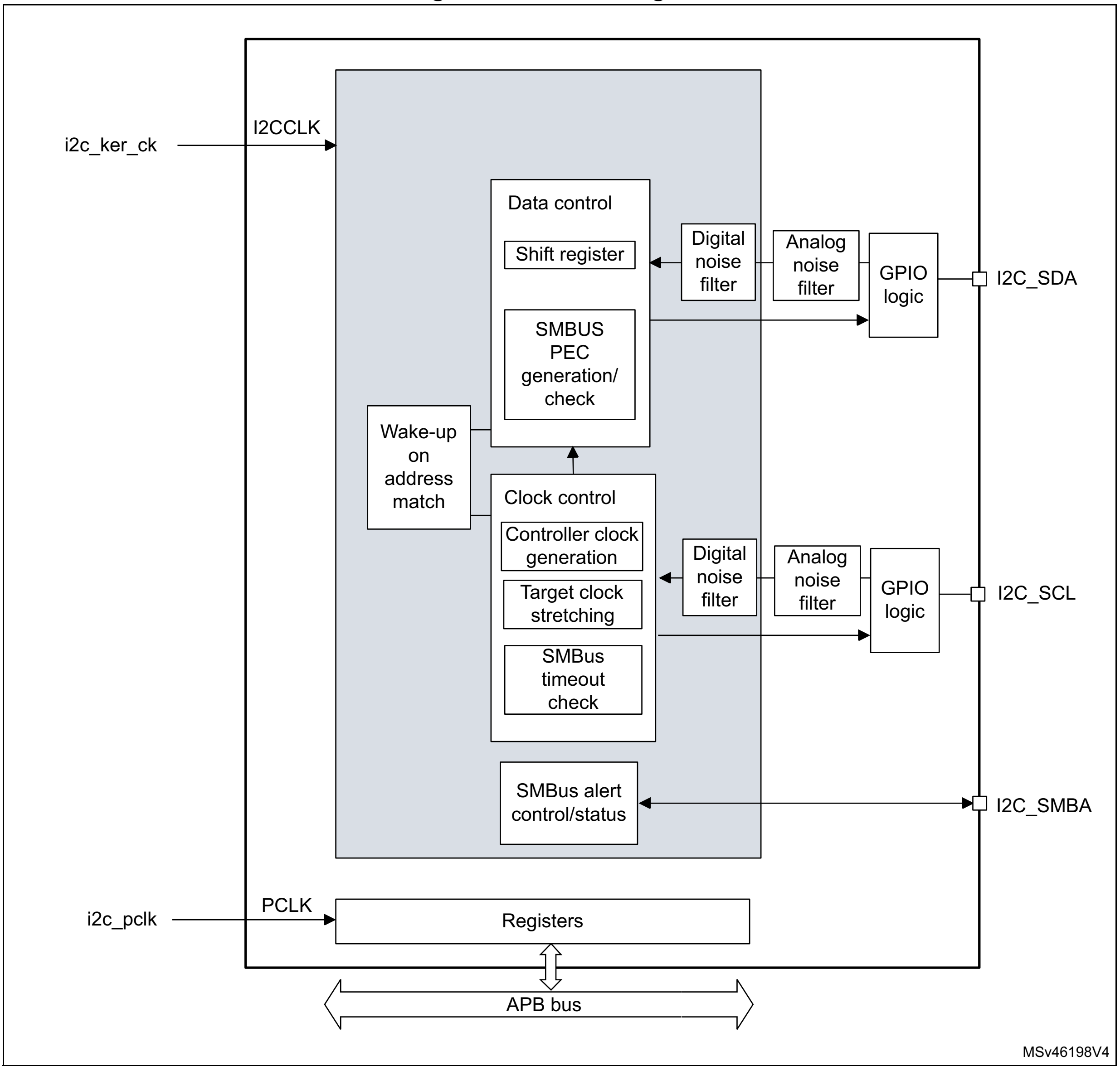
The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

For I2C I/Os supporting 20 mA output current drive for Fast-mode Plus operation, the driving capability is enabled through control bits in the system configuration block(SYSCFG).

50.4.1 I2C block diagram

Figure 514. Block diagram



50.4.2 I2C pins and internal signals

Table 346. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 347. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 361* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

50.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

50.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

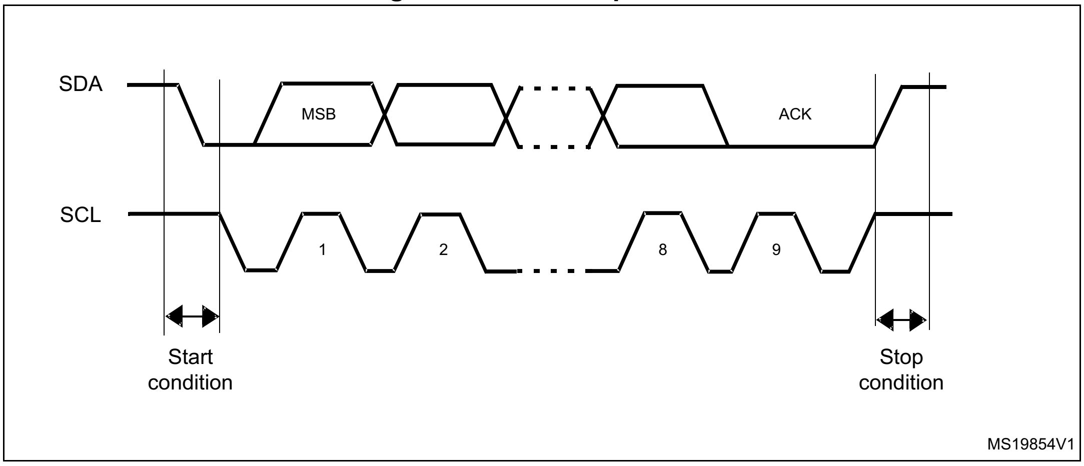
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 515. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

50.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 50.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 348. Comparison of analog and digital filters

Item	**Analog filter**	**Digital filter**
Filtering capacity[1]	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

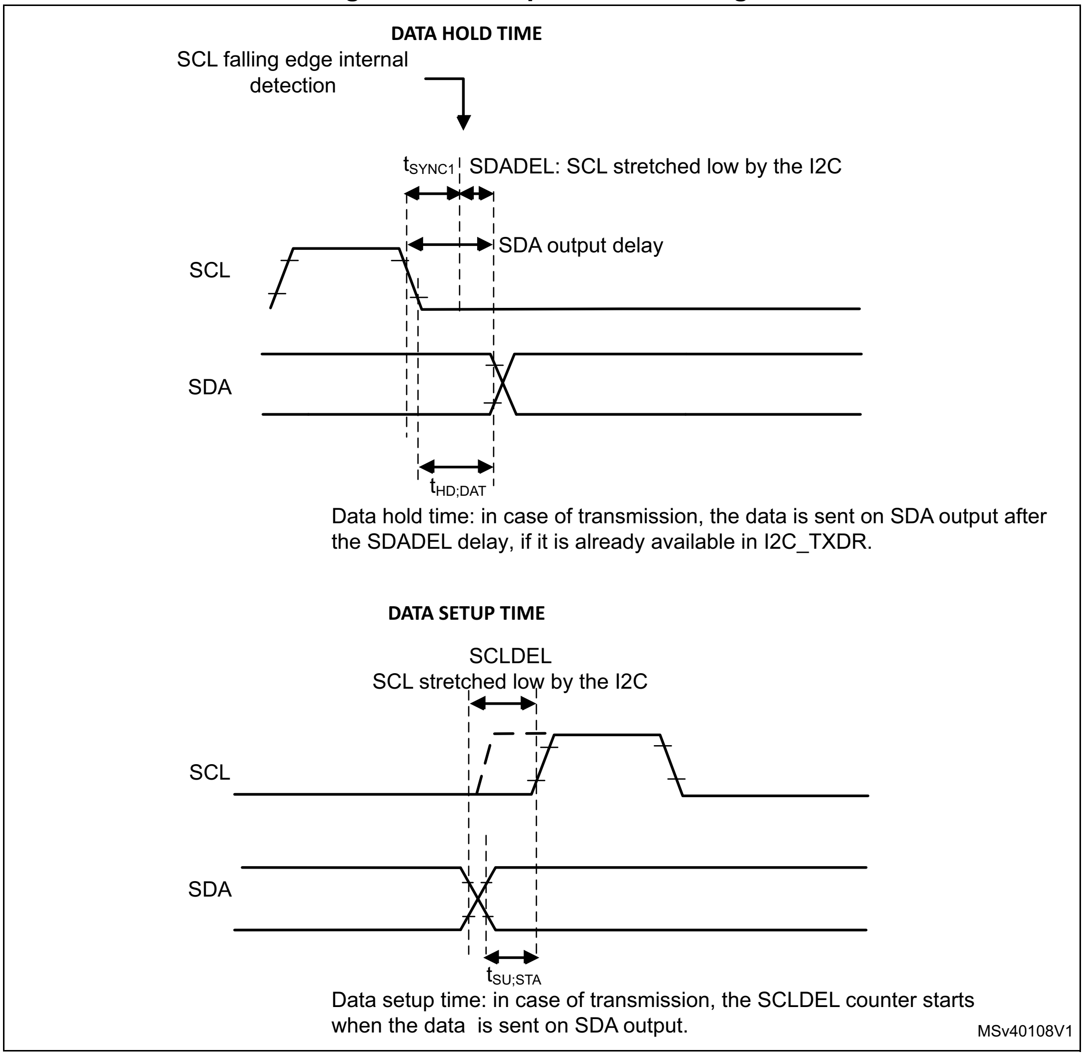
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 516. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

{$t_{f(max)}$ + $t_{HD;DAT(min)}$ - $t_{AF(min)}$ - [(DNF + 3) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}} ≤ SDADEL

SDADEL ≤ {$t_{HD;DAT\ (max)}$ - $t_{AF(max)}$ - [(DNF + 4) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}}

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

SDADEL ≤ $\{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq$ SCLDEL

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 349. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

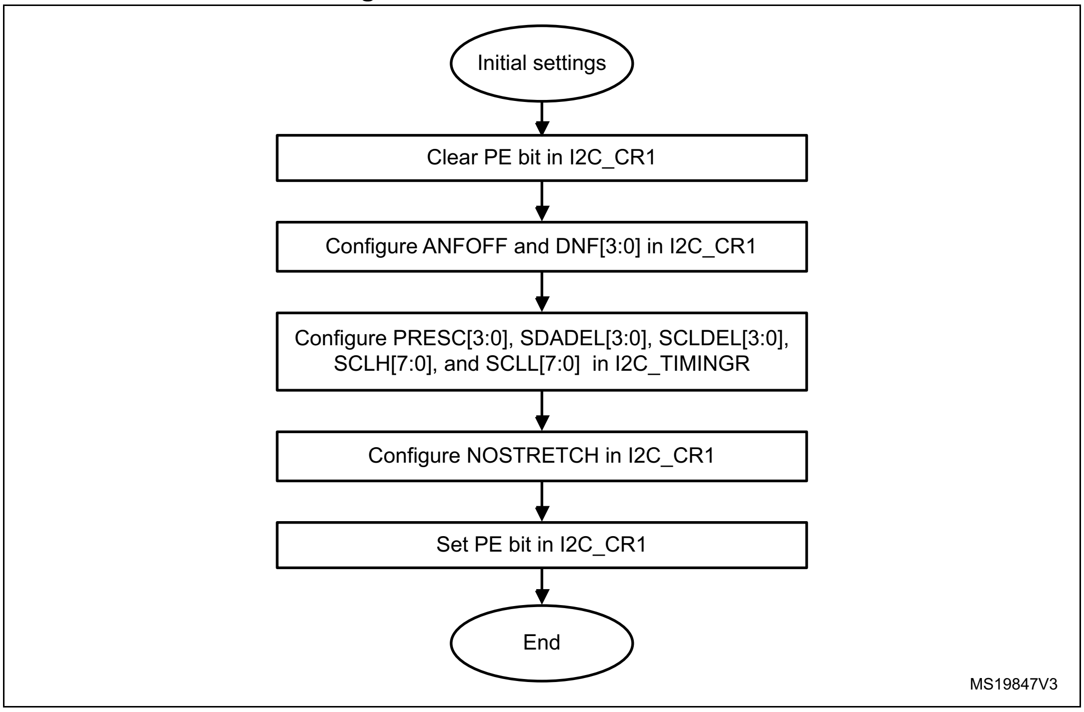
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 517. I2C initialization flow



50.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

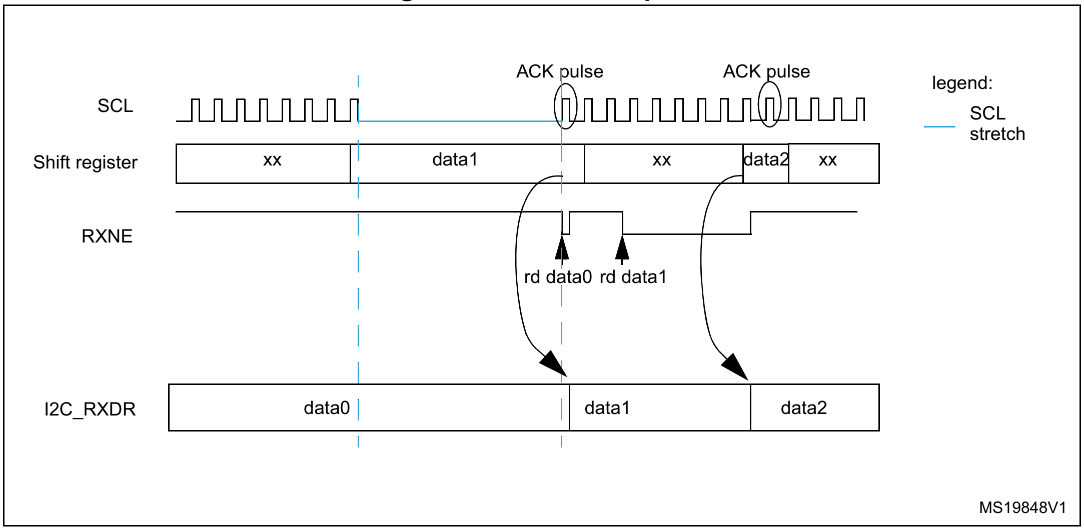
50.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

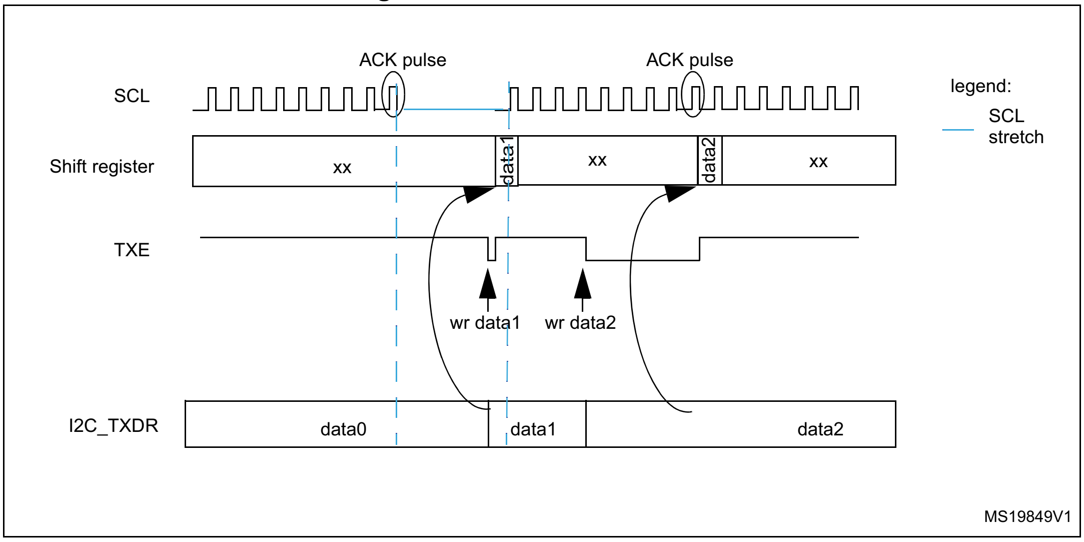
Figure 518. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 519. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 350. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

50.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

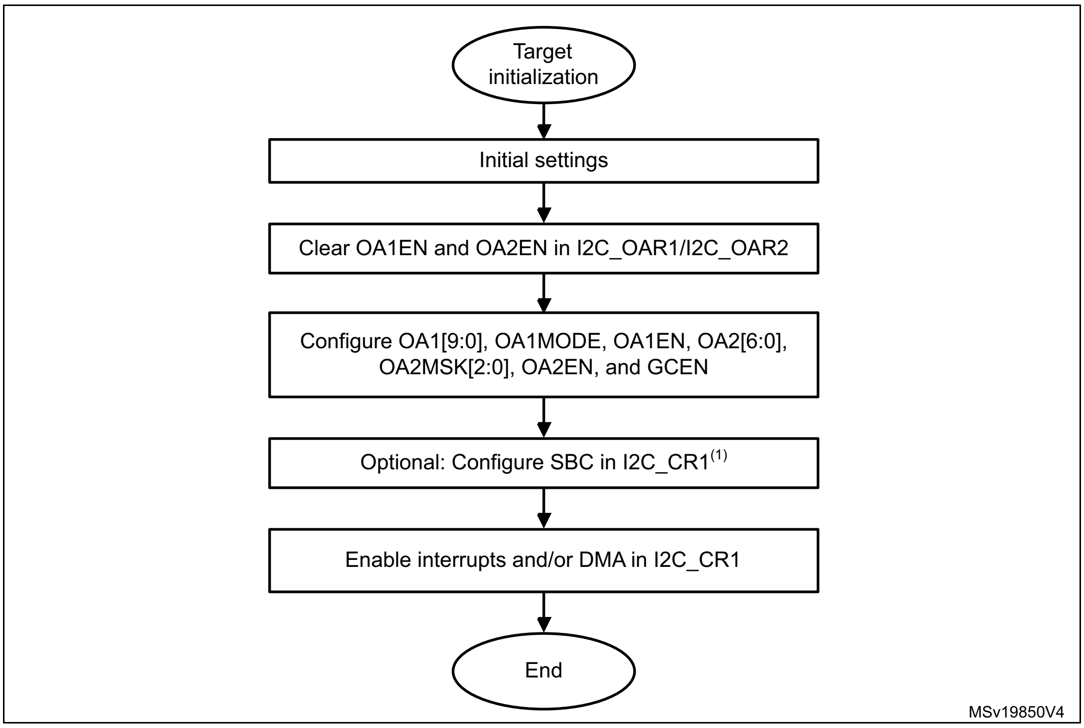
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 520. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 521. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

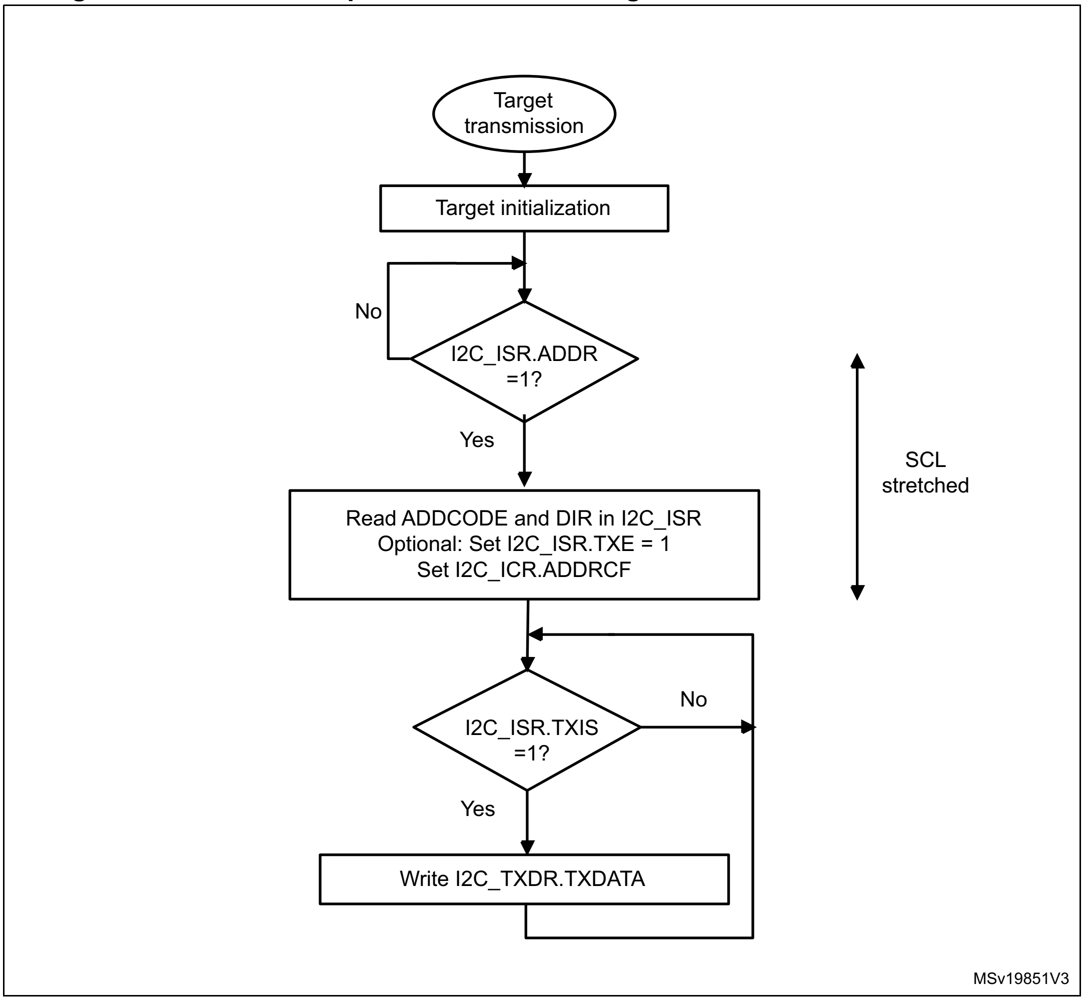


Figure 522. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

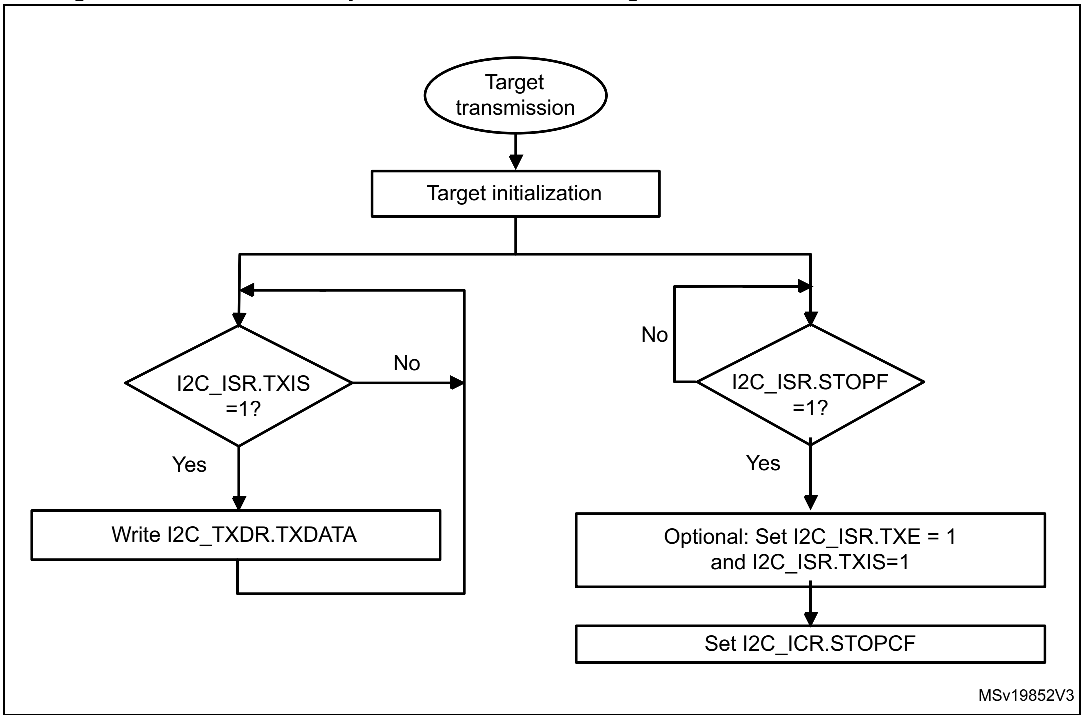
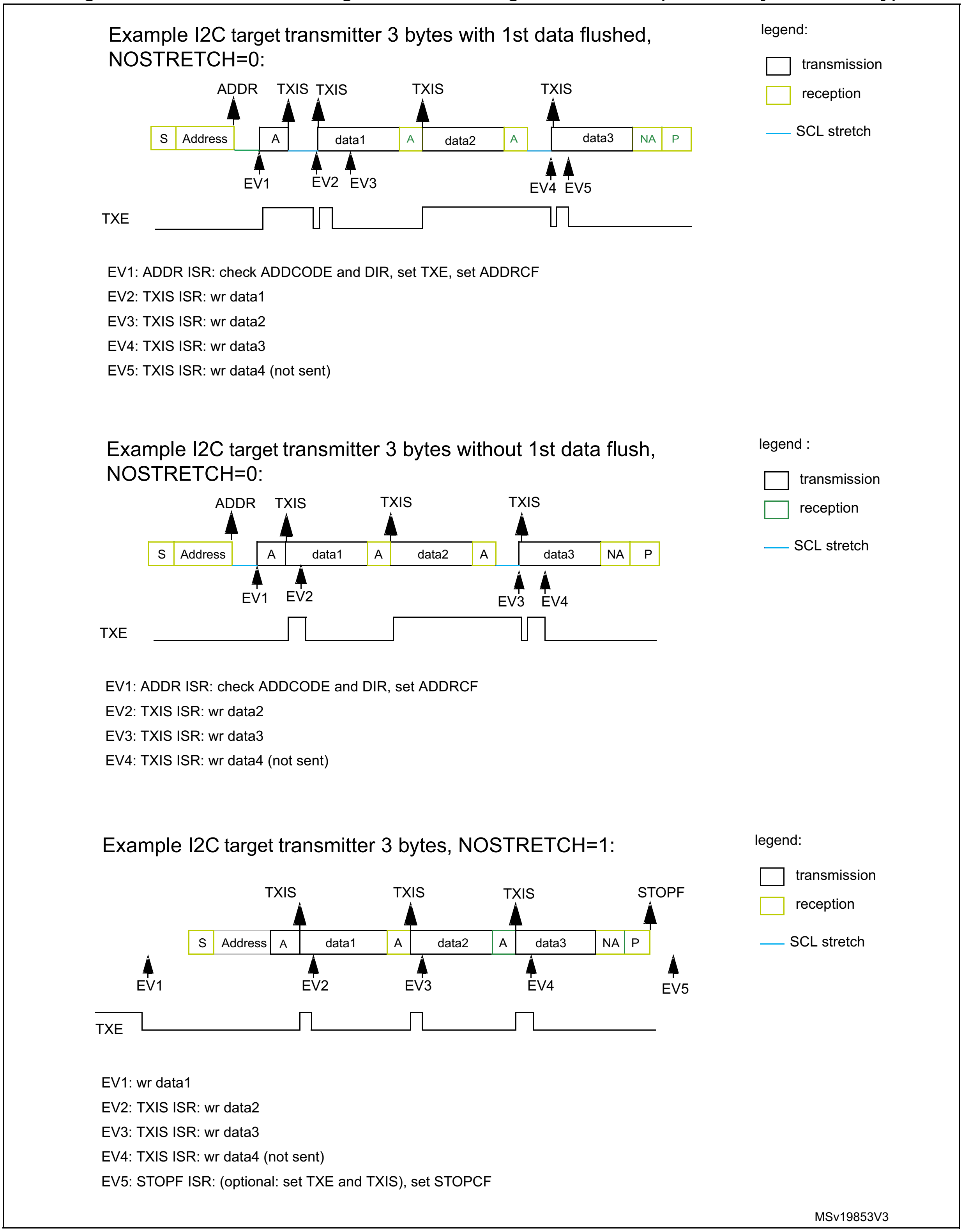


Figure 523. Transfer bus diagrams for I2C target transmitter (mandatory events only)



Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 524. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

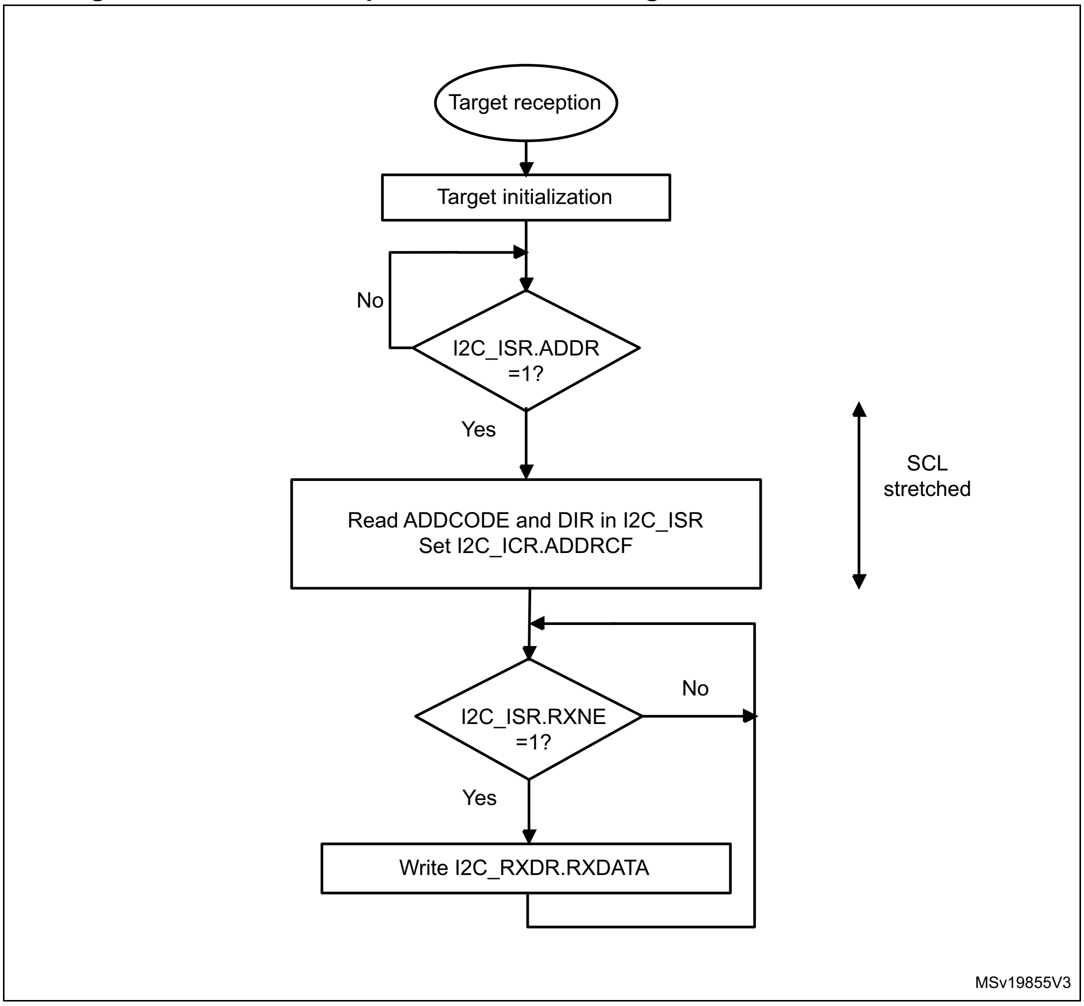


Figure 525. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

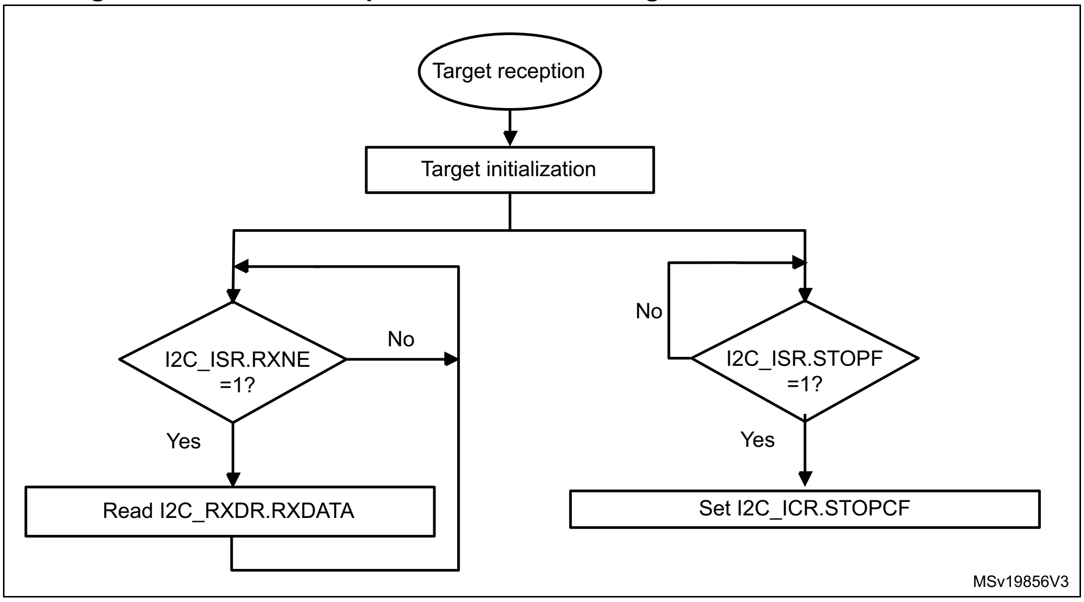
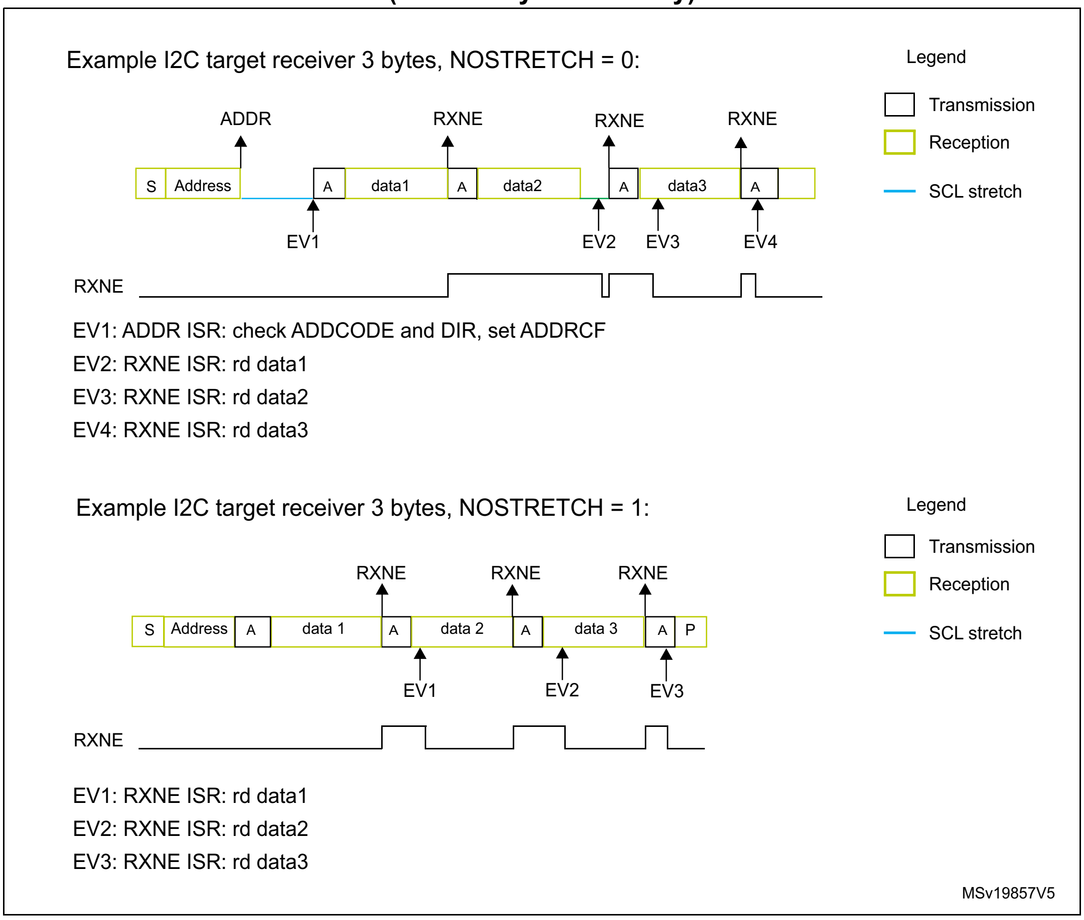


Figure 526. Transfer bus diagrams for I2C target receiver (mandatory events only)



50.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

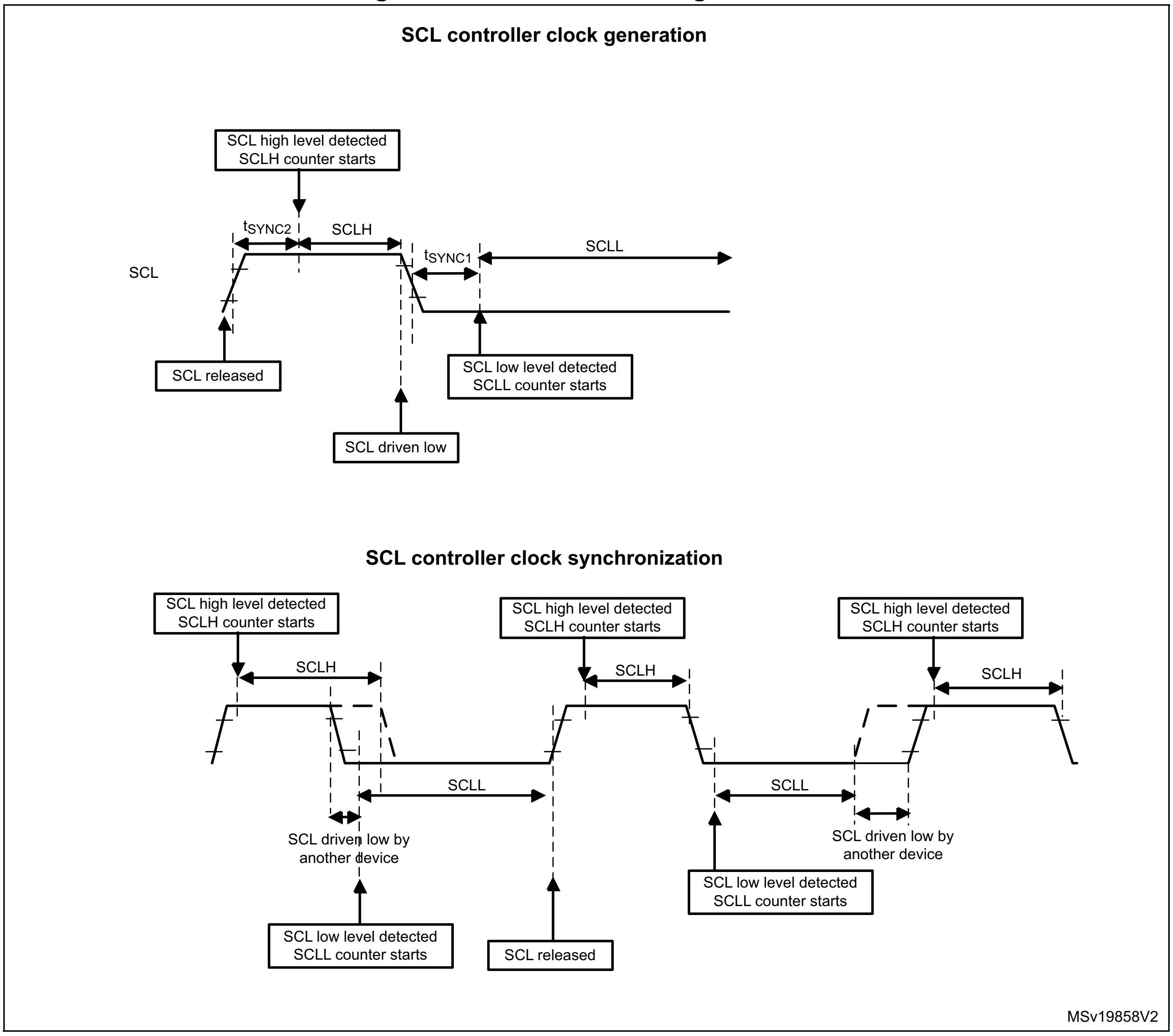
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 527. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 351. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 50.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

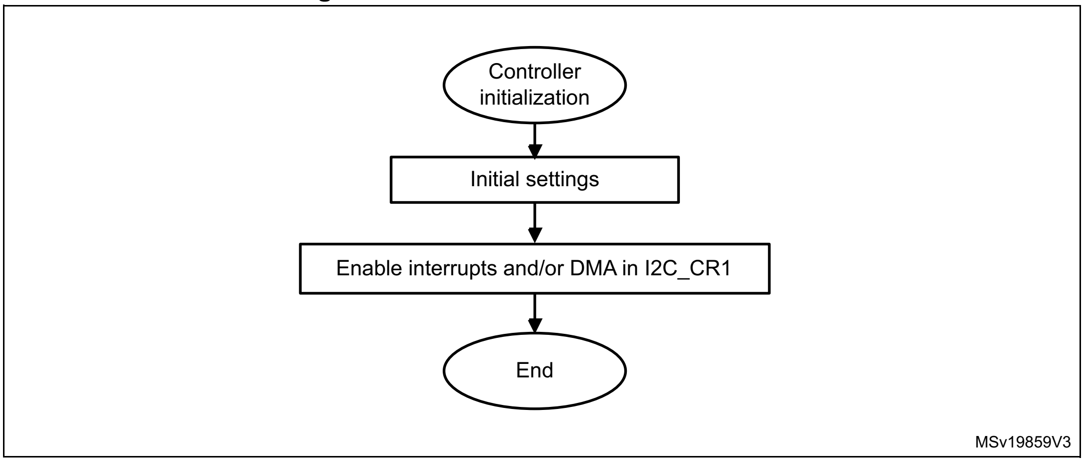
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared when the ADDRCF bit is set.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 528. Controller initialization flow

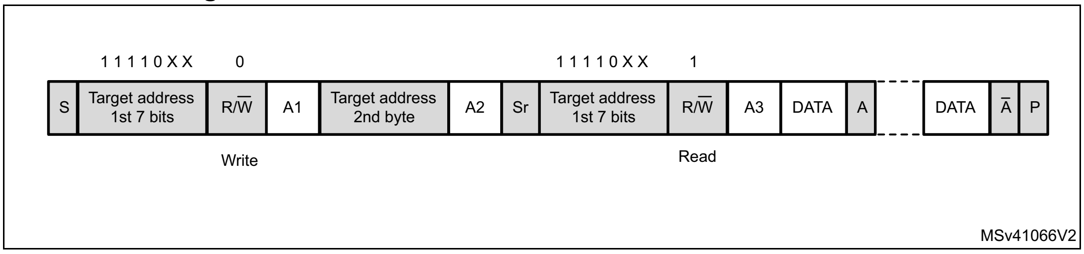


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

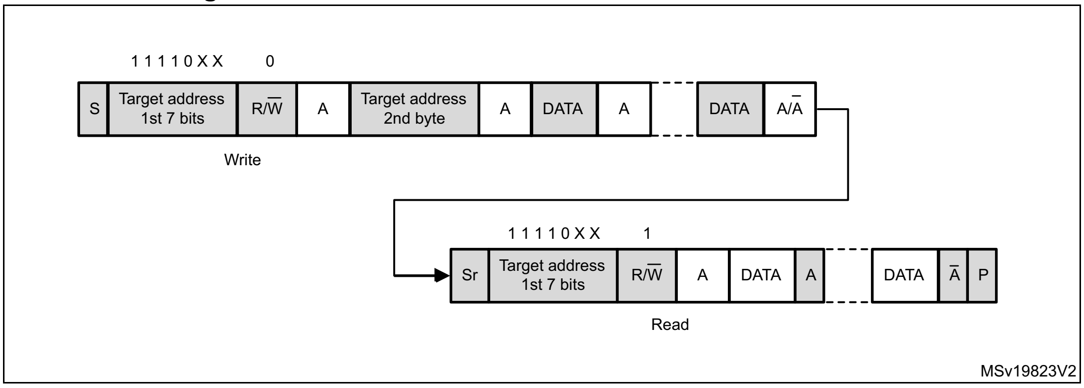
Figure 529. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 530. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 531. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

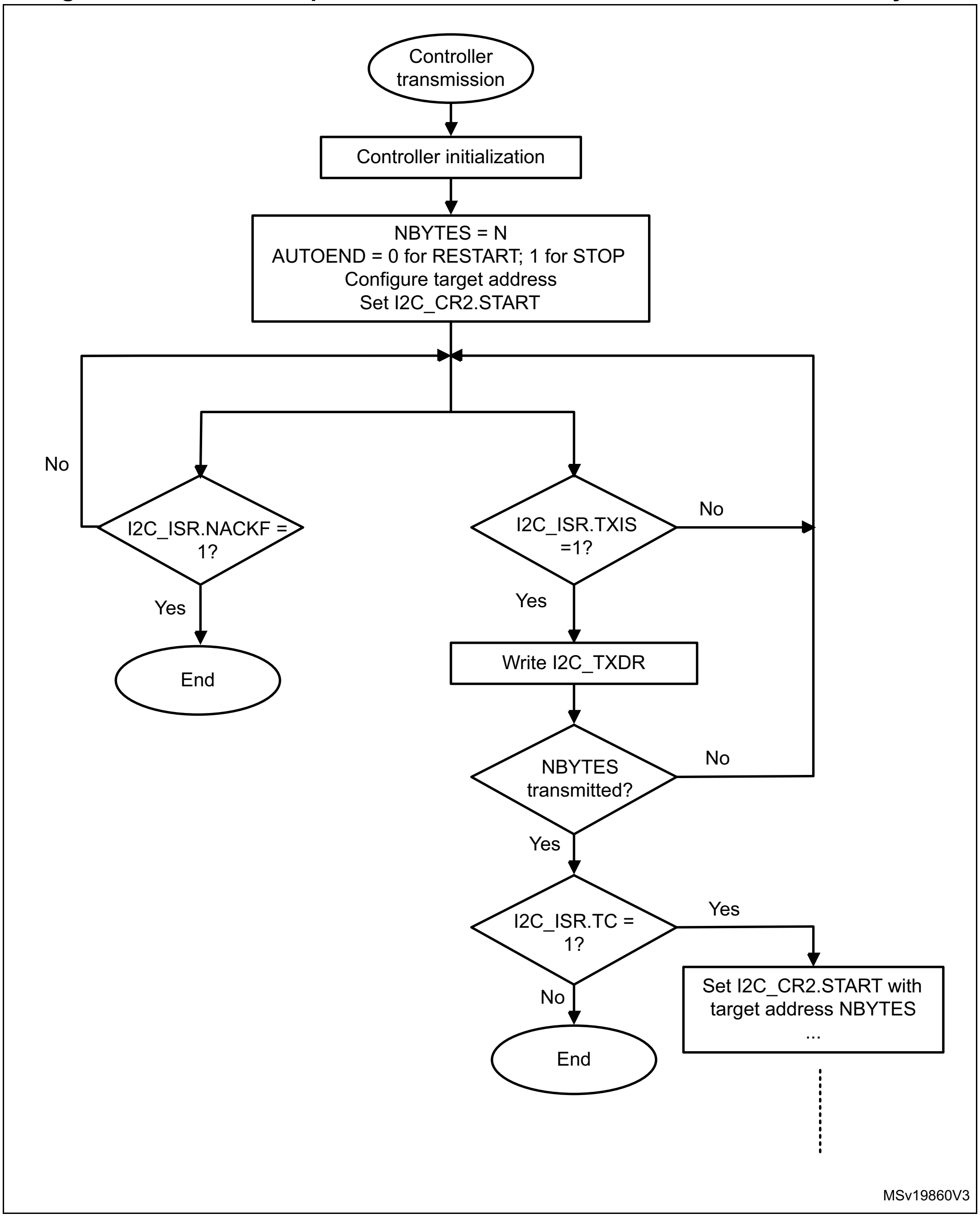


Figure 532. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

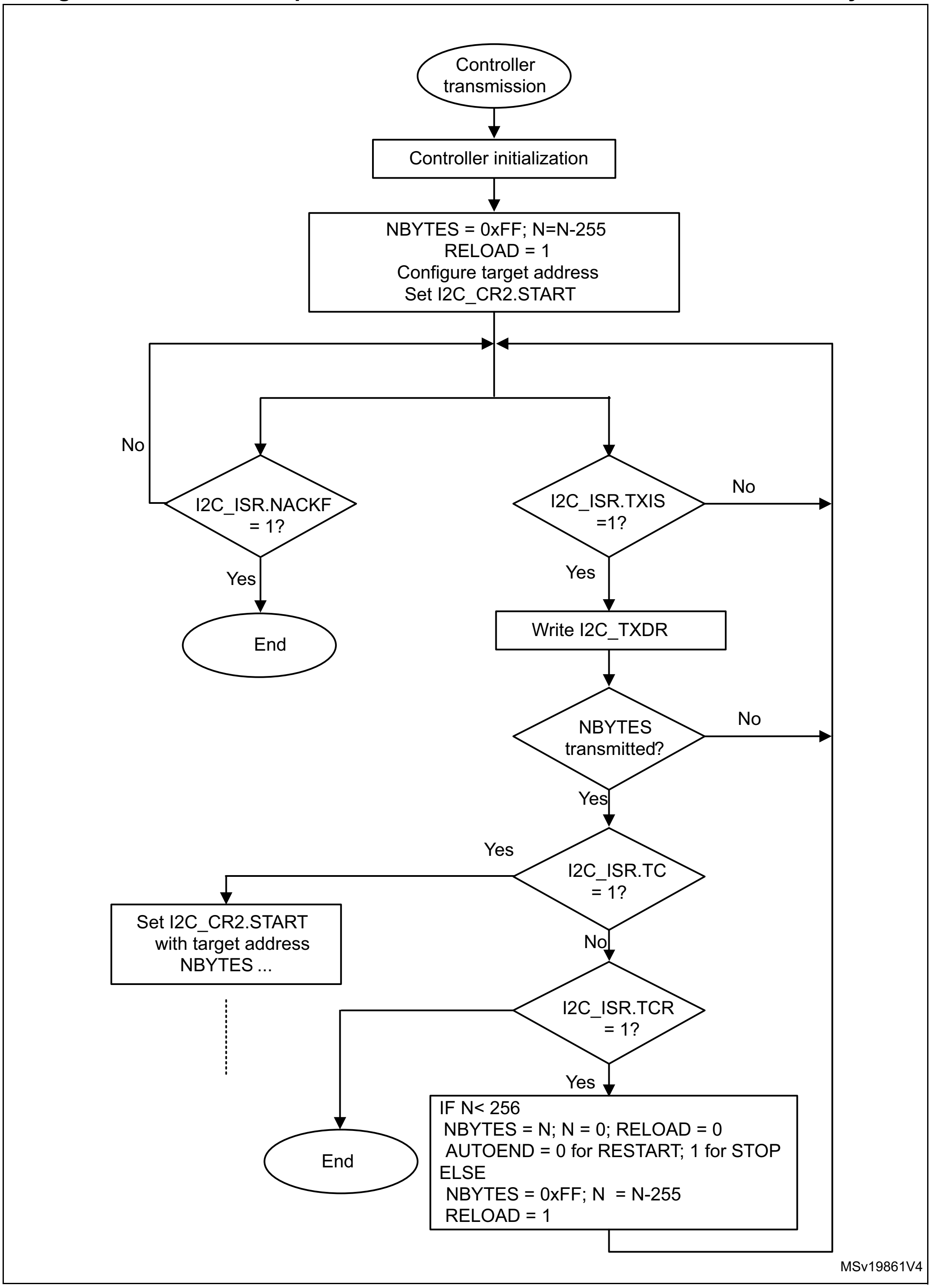
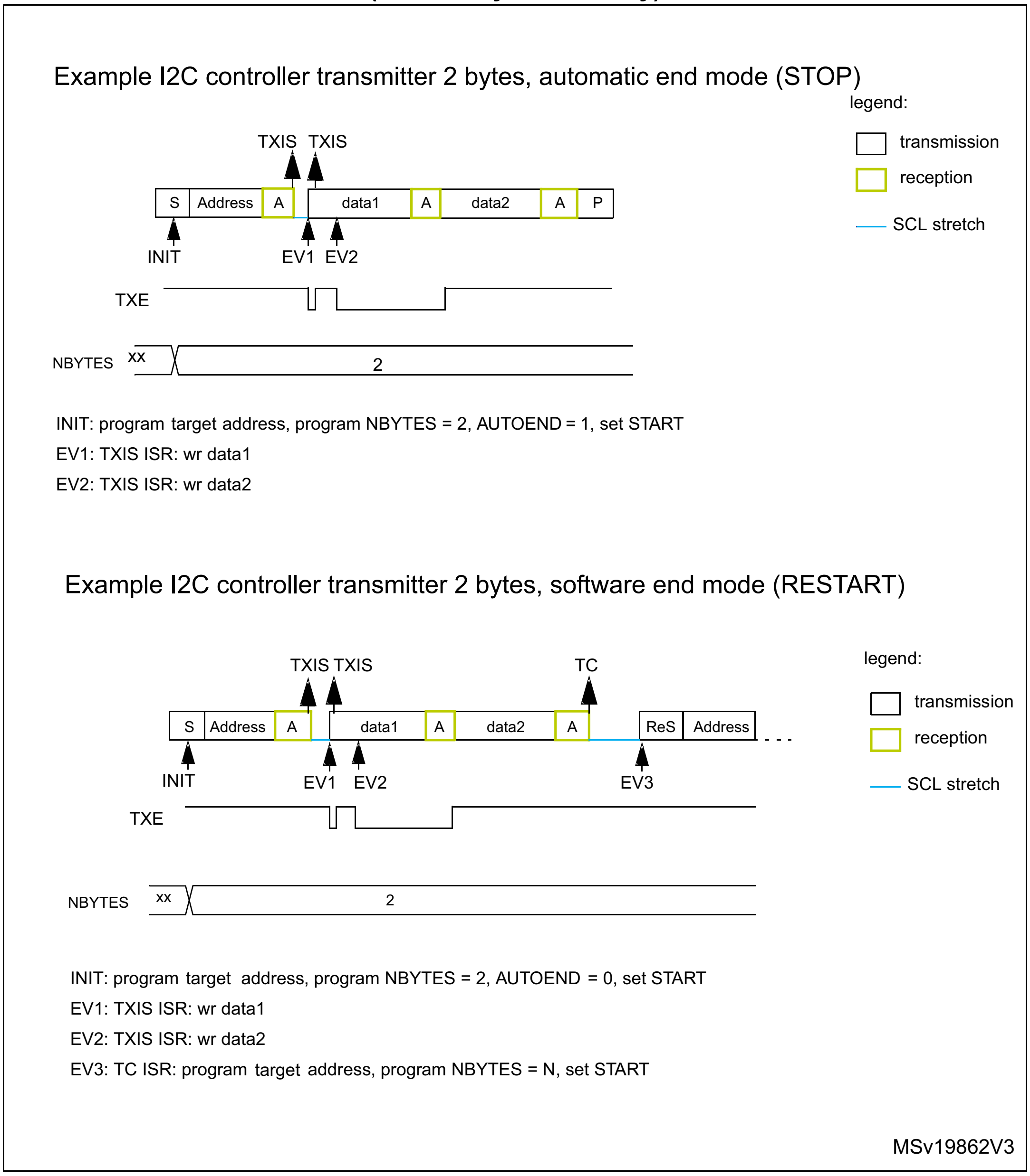


Figure 533. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 534. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

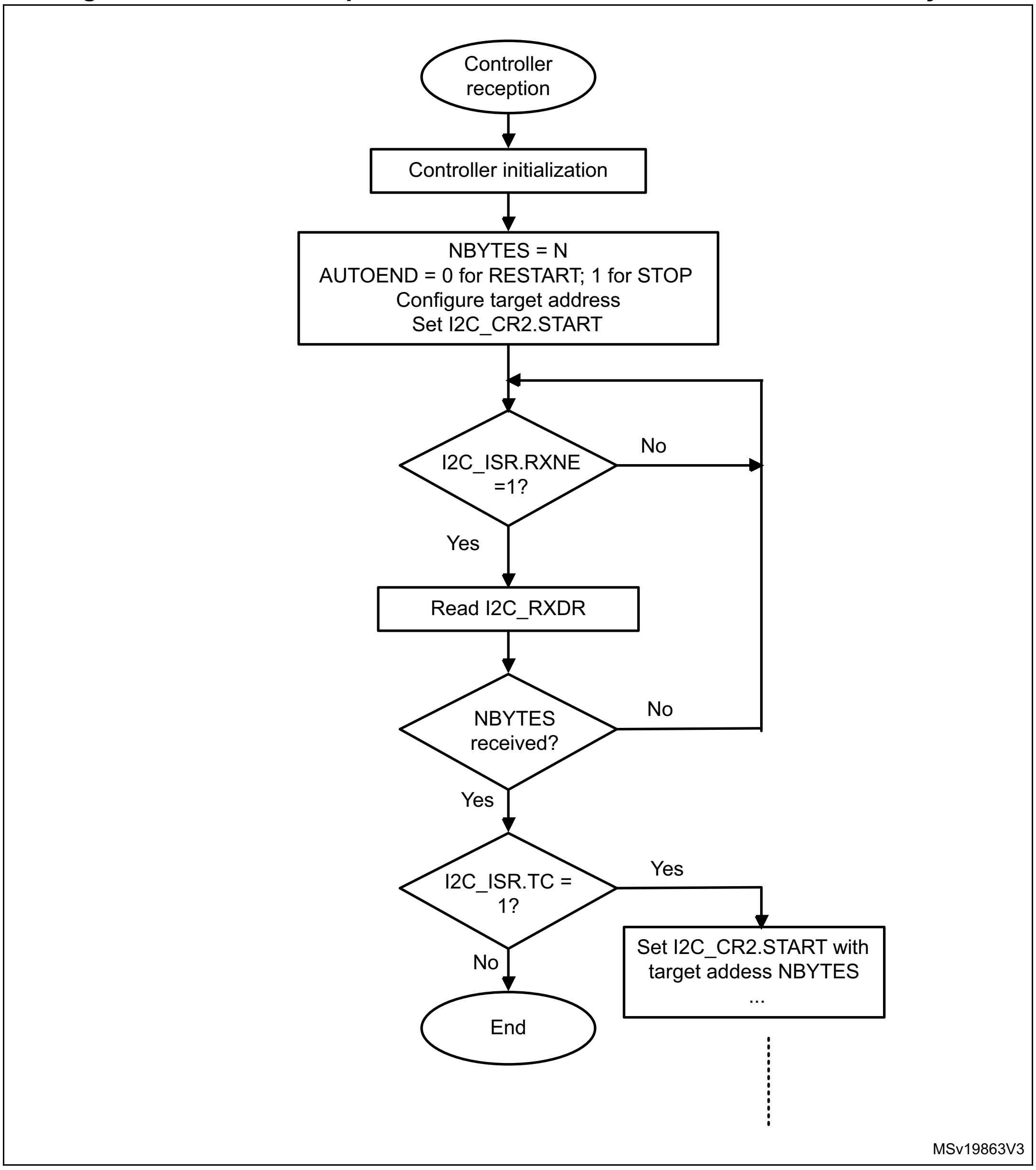


Figure 535. Transfer sequence flow for I2C controller receiver, N > 255 bytes

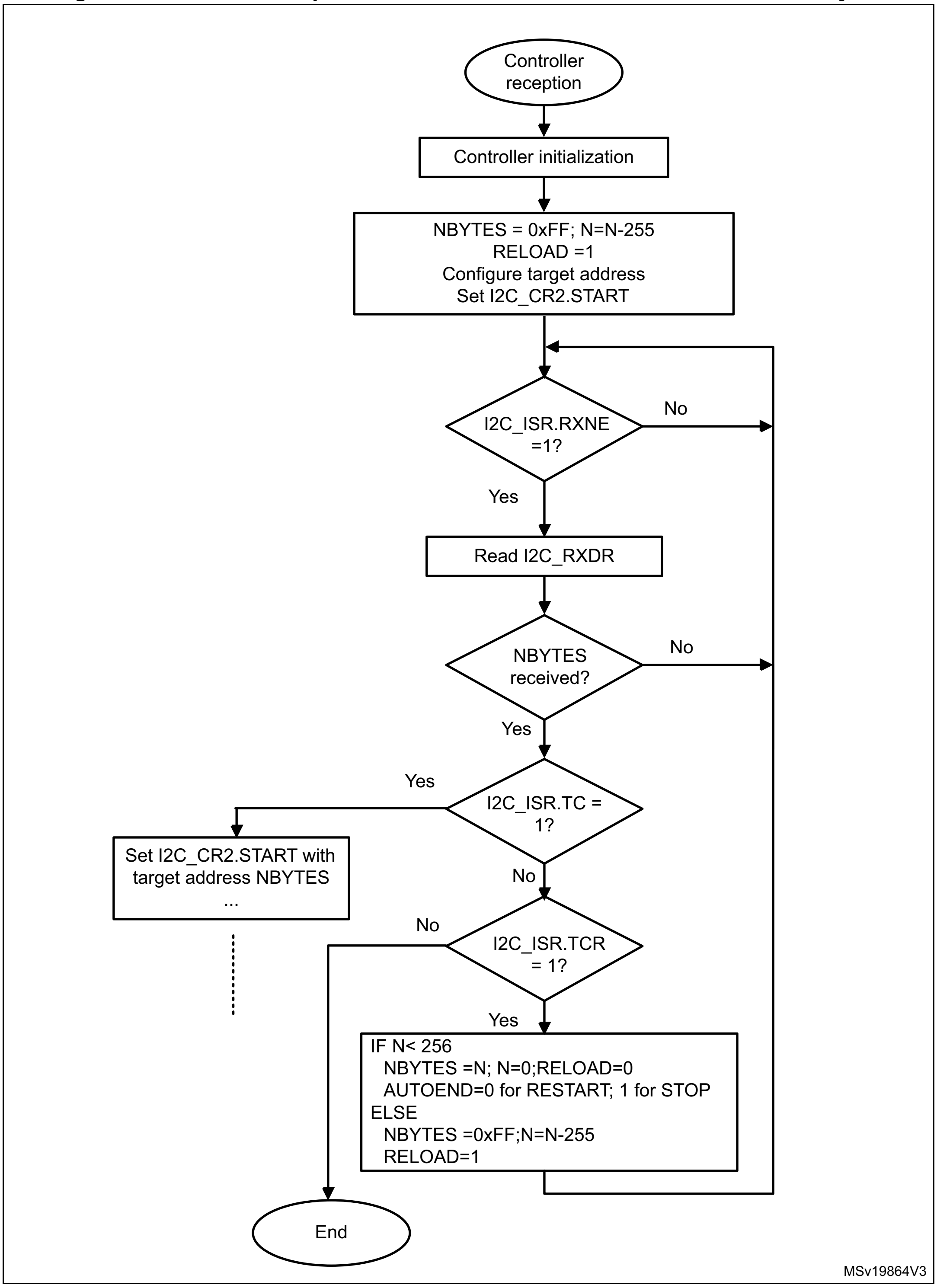
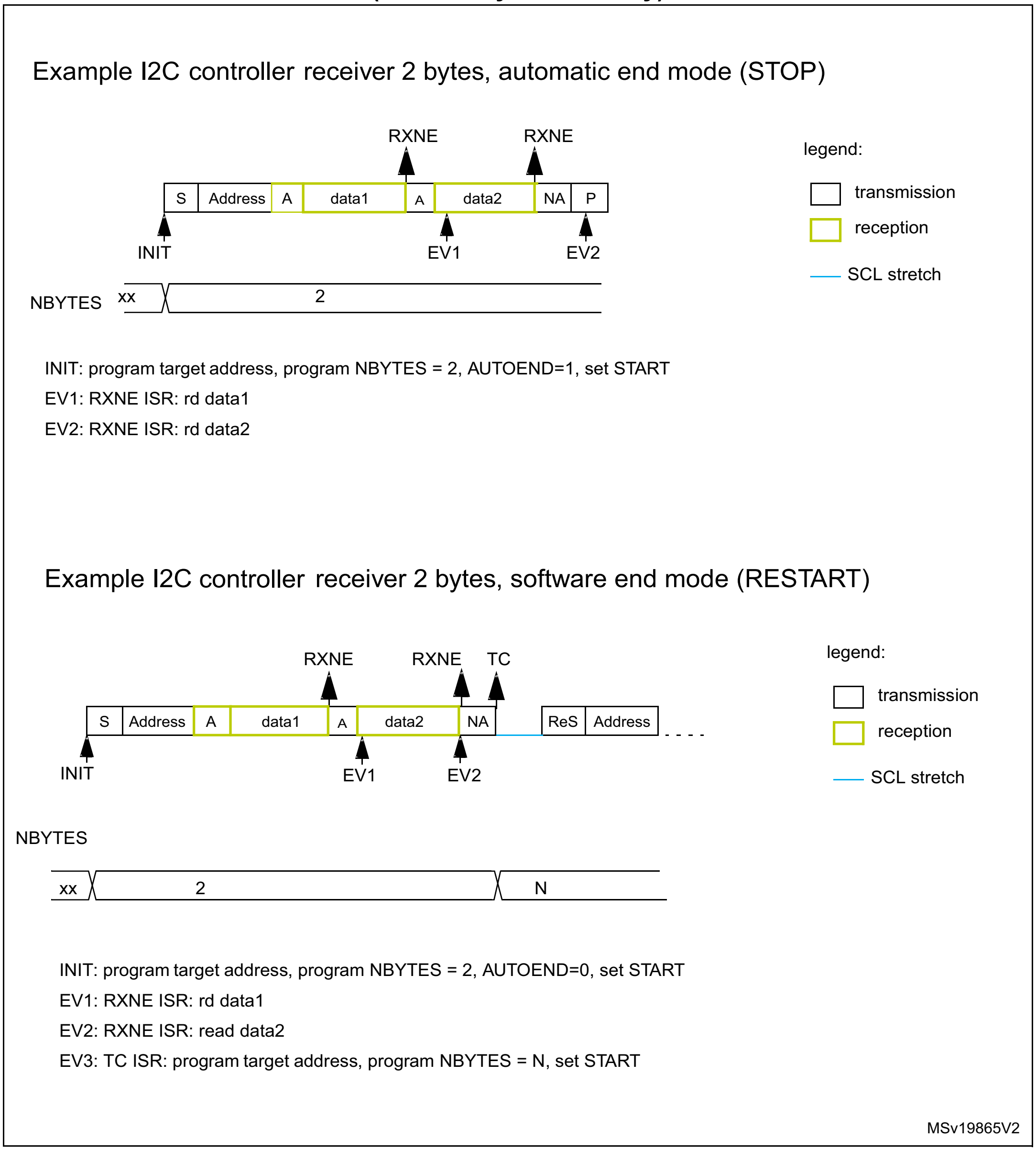


Figure 536. Transfer bus diagrams for I2C controller receiver (mandatory events only)



50.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 352. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 353. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

Table 354. Timing settings for f_{I2CCLK} of 48 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0xB	0xB	0x5	0x5
SCLL[7:0]	0xC7	0x13	0x9	0x3
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	4 x 125 ns = 500 ns
SCLH[7:0]	0xC3	0xF	0x3	0x1
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~875 ns(4)
SDADEL[3:0]	0x2	0x2	0x3	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	3 x 125 ns = 375 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to the SCL internal detection delay. Values provided for t_{SCL} are only examples.
2. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 1000 ns
3. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 750 ns
4. $t_{SYNC1} + t_{SYNC2}$ minimum value is 4x t_{I2CCLK} = 83.3 ns. Example with $t_{SYNC1} + t_{SYNC2}$ = 250 ns

50.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

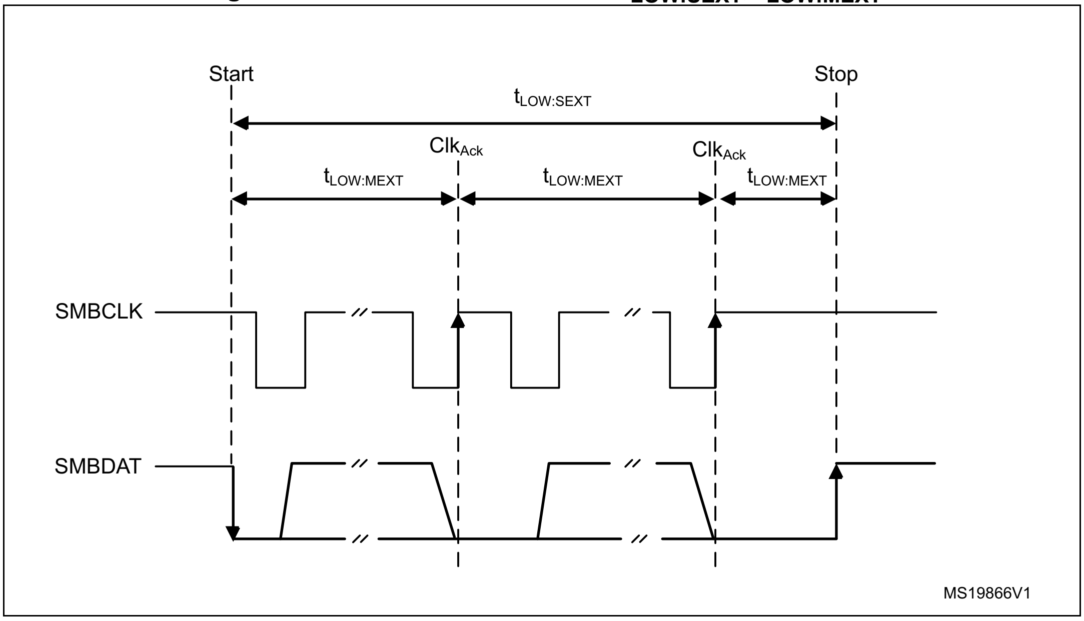
Table 355. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 537. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}(max)$ (refer to the table in *Section 50.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

50.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 356. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 357*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 358*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 359*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

50.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 357. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms
48 MHz	0x249	0	1	586 x 2048 x 20.08 ns = 25 ms

Table 358. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms
48 MHz	0xBB	1	188 x 2048 x 20.08 ns = 8 ms

Table 359. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs
48 MHz	0x257	1	1	600 x 4 x 20.08 ns = 50 µs

50.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 50.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 538. Transfer sequence flow for SMBus target transmitter N bytes + PEC

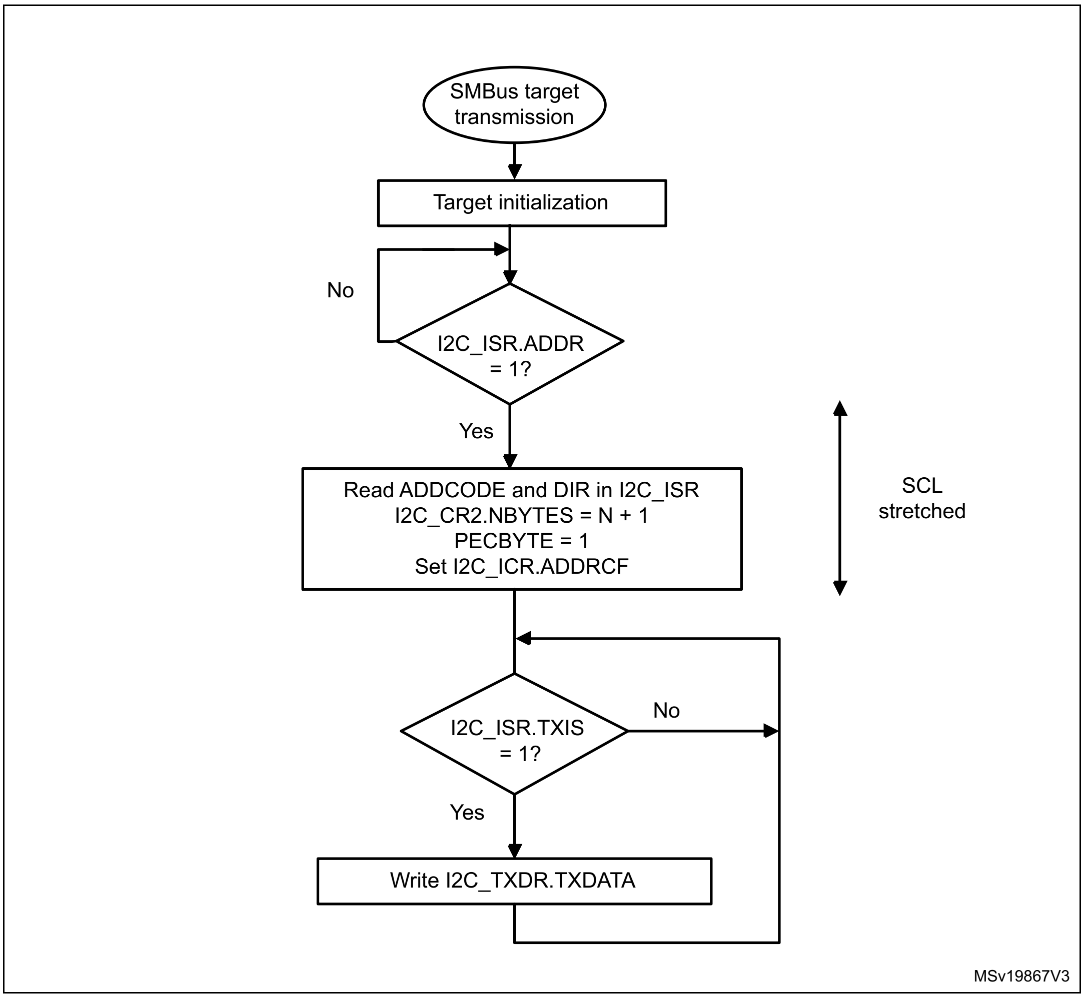
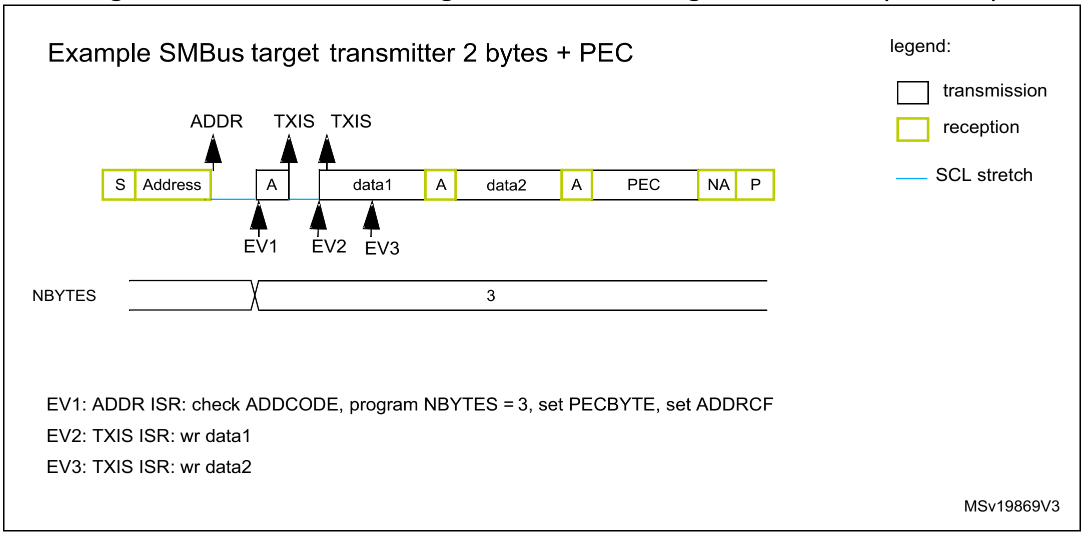


Figure 539. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 540. Transfer sequence flow for SMBus target receiver N bytes + PEC

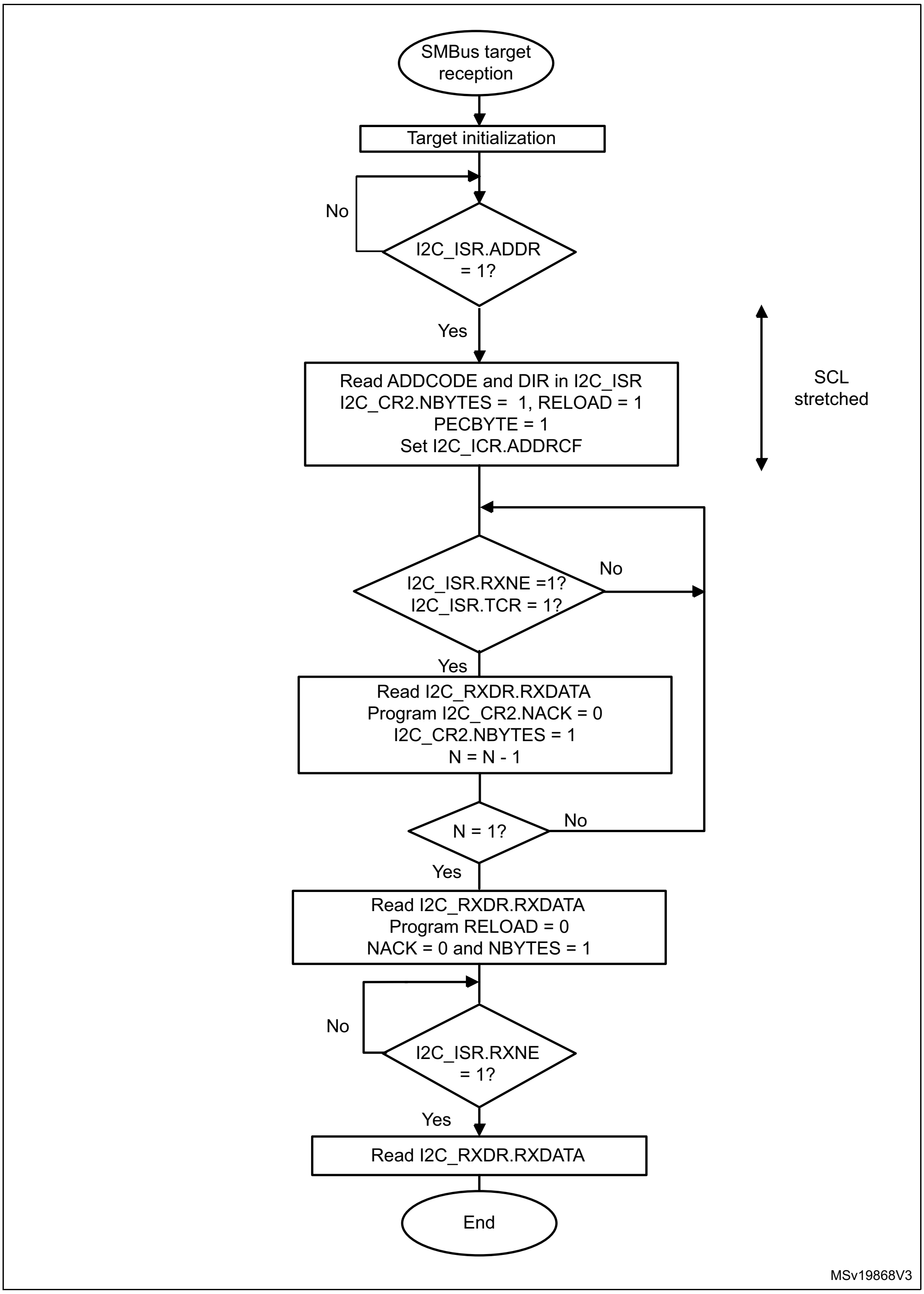
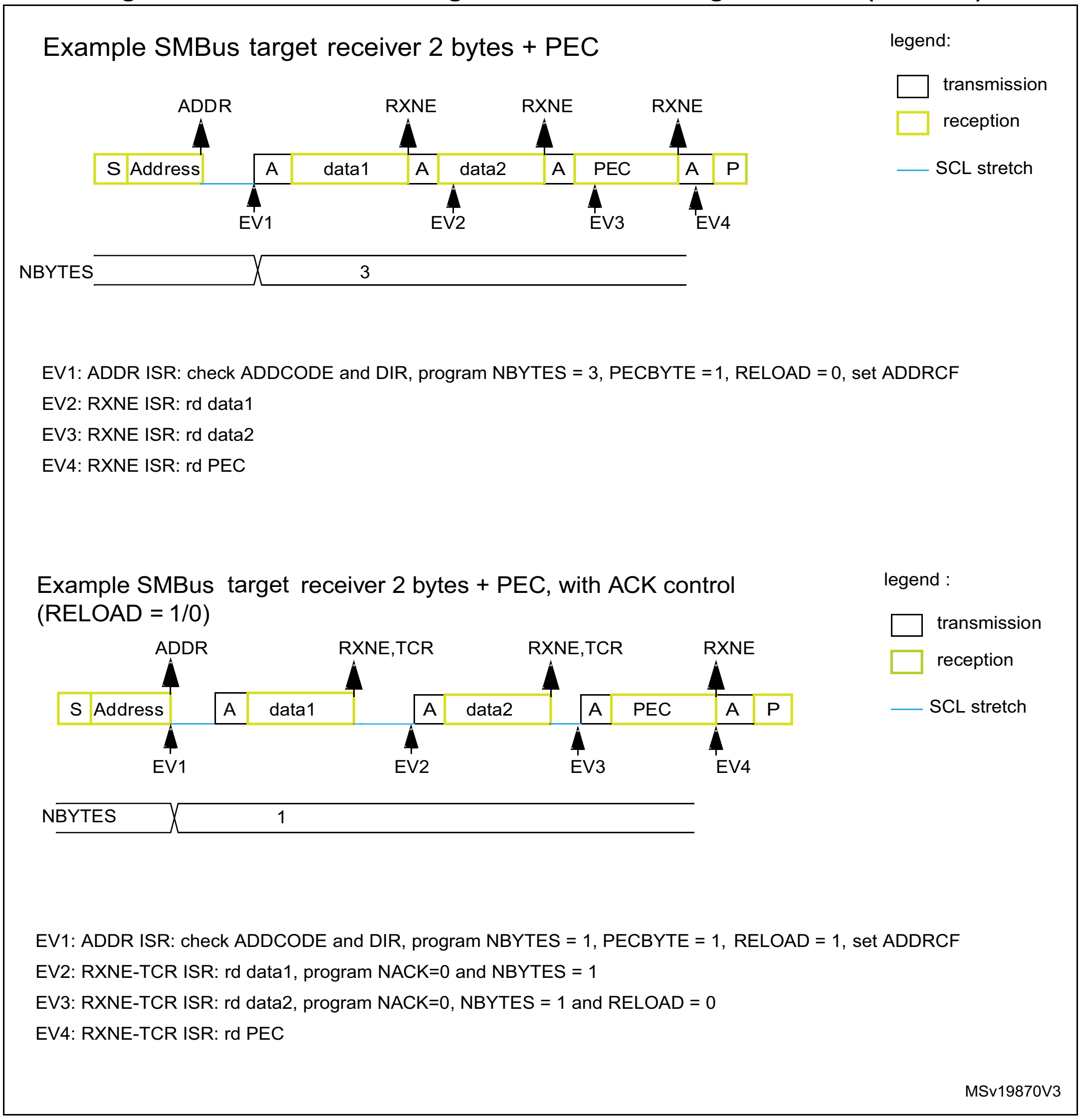


Figure 541. Bus transfer diagrams for SMBus target receiver (SBC = 1)



50.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 50.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

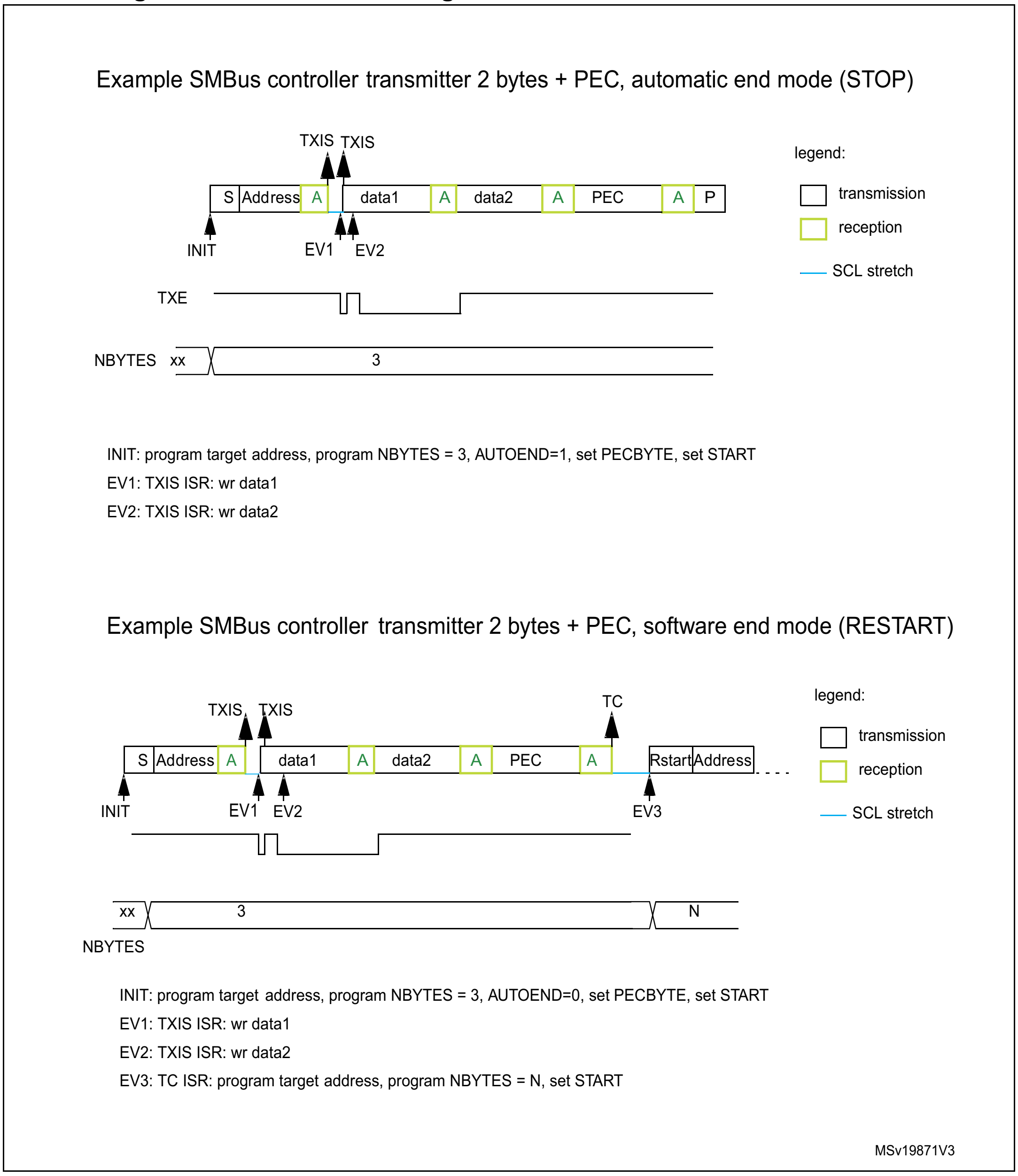
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 542. Bus transfer diagrams for SMBus controller transmitter



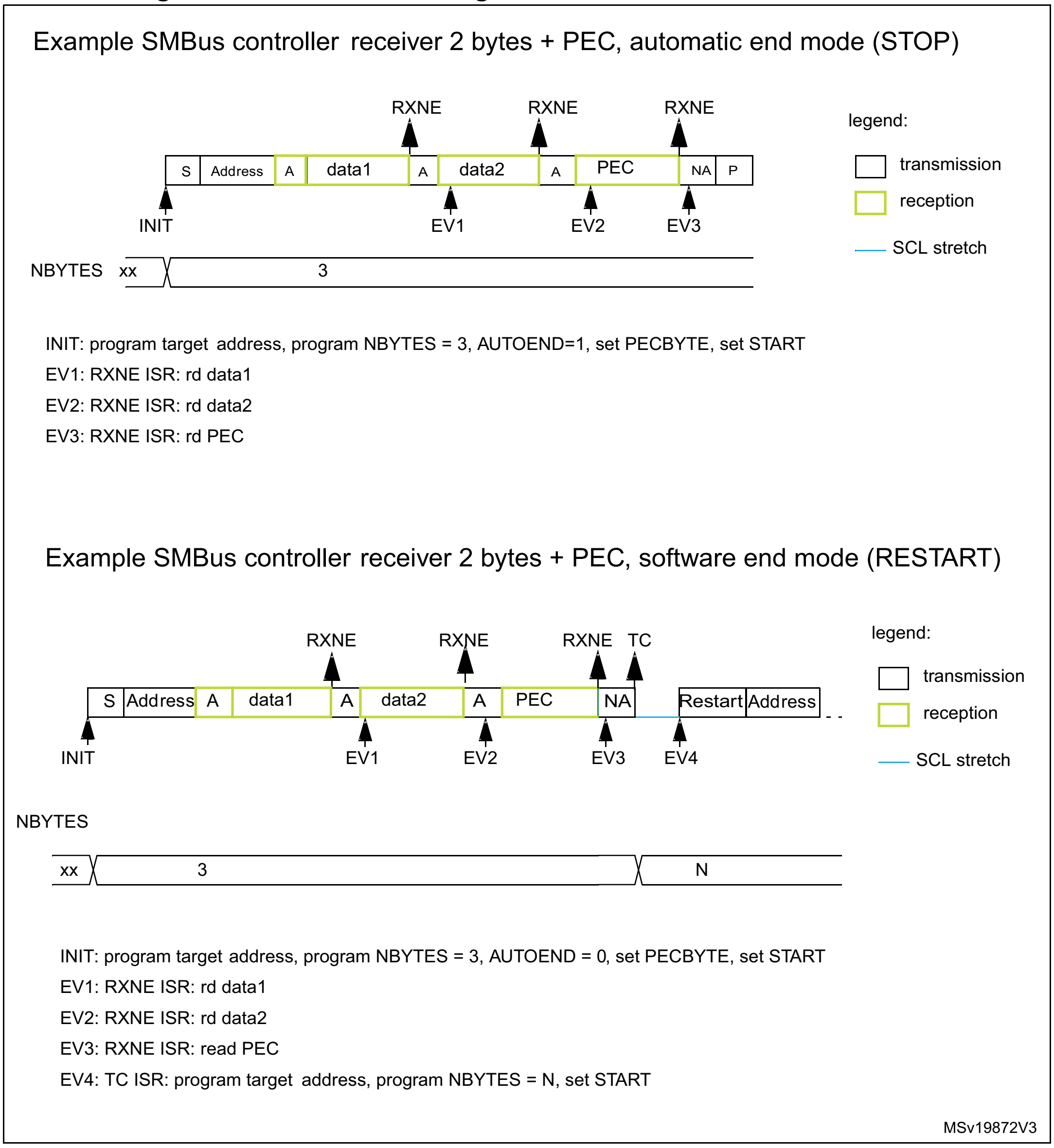
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 543. Bus transfer diagrams for SMBus controller receiver



50.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI or CSI oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI or CSI oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI or CSI oscillator and stretches SCL low until the oscillator wakes up.

HSI or CSI is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI or CSI oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI or CSI oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI or CSI oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

50.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

50.5 I2C in low-power modes

Table 360. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI or CSI). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 50.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

50.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 361. I2C interrupt requests

<table>
<tr><th>Interrupt acronym</th><th>Interrupt event</th><th>Event flag</th><th>Enable control bit</th><th>Interrupt clear method</th><th>Exit Sleep mode</th><th>Exit Stop modes</th><th>Exit Standby modes</th></tr>
<tr><td rowspan="7">I2C_EV</td><td>Receive buffer not empty</td><td>RXNE</td><td>RXIE</td><td>Read I2C_RXDR register</td><td rowspan="7">Yes</td><td rowspan="5">No</td><td rowspan="7">No</td></tr>
<tr><td>Transmit buffer interrupt status</td><td>TXIS</td><td>TXIE</td><td>Write I2C_TXDR register</td></tr>
<tr><td>STOP detection interrupt flag</td><td>STOPF</td><td>STOPIE</td><td>Write STOPCF = 1</td></tr>
<tr><td>Transfer complete reload</td><td>TCR</td><td rowspan="2">TCIE</td><td>Write I2C_CR2 with NBYTES[7:0] ≠ 0</td></tr>
<tr><td>Transfer complete</td><td>TC</td><td>Write START = 1 or STOP = 1</td></tr>
<tr><td>Address matched</td><td>ADDR</td><td>ADDRIE</td><td>Write ADDRCF = 1</td><td>Yes(1)</td></tr>
<tr><td>NACK reception</td><td>NACKF</td><td>NACKIE</td><td>Write NACKCF = 1</td><td>No</td></tr>
<tr><td rowspan="3">I2C_ERR</td><td>Bus error</td><td>BERR</td><td rowspan="3">ERRIE</td><td>Write BERRCF = 1</td><td rowspan="3">Yes</td><td rowspan="3">No</td><td rowspan="3">No</td></tr>
<tr><td>Arbitration loss</td><td>ARLO</td><td>Write ARLOCF = 1</td></tr>
<tr><td>Overrun/underrun</td><td>OVR</td><td>Write OVRCF = 1</td></tr>
<tr><td rowspan="3">I2C_ERR</td><td>PEC error</td><td>PECERR</td><td rowspan="3">ERRIE</td><td>Write PECERRCF = 1</td><td rowspan="3">Yes</td><td rowspan="3">No</td><td rowspan="3">No</td></tr>
<tr><td>Timeout/ t_{LOW} error</td><td>TIMEOUT</td><td>Write TIMEOUTCF = 1</td></tr>
<tr><td>SMBus alert</td><td>ALERT</td><td>Write ALERTCF = 1</td></tr>
</table>

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 50.3: I2C implementation*.

50.7 I2C DMA requests

50.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 18: Direct memory access controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

50.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 18: Direct memory access controller (DMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

50.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the I2Cx bits in the DBG block.

50.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

50.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to
2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **PECEN:** PEC enable
- 0: PEC calculation disabled
- 1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable
- 0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
- 1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable
- 0: Device default address disabled. Address 0b1100001x is NACKed.
- 1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable
- 0: Host address disabled. Address 0b0001000x is NACKed.
- 1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable
- 0: General call disabled. Address 0b00000000 is NACKed.
- 1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable
- 0: Wake-up from Stop mode disabled.
- 1: Wake-up from Stop mode enabled.

Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.

0: Clock stretching enabled

1: Clock stretching disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control

This bit is used to enable hardware byte control in target mode.

0: Target byte control disabled

1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception

1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission

1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF

0: Analog noise filter enabled

1: Analog noise filter disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}

...

1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}

Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

Note: Any of these errors generates an interrupt:

- arbitration loss (ARLO)

- bus error detection (BERR)

- overrun/underrun (OVR)

- timeout detection (TIMEOUT)

- PEC error detection (PECERR)

- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable

0: Transfer complete interrupt disabled

1: Transfer complete interrupt enabled

Note: Any of these events generates an interrupt:

Transfer complete (TC)

Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disabled
1: Peripheral enabled

Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

50.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte
This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.
0: No PEC transfer
1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.
This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by a timeout error detection, or when PE = 0. It can also be cleared by software, by setting the ADDRCF bit of the I2C_ICR register.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):

SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):

SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

50.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.

1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.

1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

50.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable

0: Own address 2 disabled. The received target address OA2 is NACKed.

1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks

000: No mask

001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.

010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.

011: OA2[3:1] are masked and don’t care. Only OA2[7:4] are compared.

100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.

101: OA2[5:1] are masked and don’t care. Only OA2[7:6] are compared.

110: OA2[6:1] are masked and don’t care. Only OA2[7] is compared.

111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.

As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

50.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

50.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

- Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected
- Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

50.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

– as a controller, provided that the STOP condition is generated by the peripheral.

– as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

50.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear

Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear

Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear

Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear

Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear

Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear

Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear

Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear

Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear

Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

50.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register

This field contains the internal PEC when PECEN=1.

The PEC is cleared by hardware when PE = 0.

50.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data

Data byte received from the I²C-bus.

50.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

50.9.12 I2C hardware configuration register (I2C_HWCFGR)

Address offset: 0x3F0

Reset value: 0x0000 0111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.				WKP[3:0]				ASYN[3:0]				SMBUS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **WKP[3:0]**:

0: Wake-up from Stop mode not implemented

1: Wake-up from Stop mode implemented

Bits 7:4 **ASYN[3:0]**:

0: Independent kernel clock not implemented

1: Independent kernel clock implemented

Bits 3:0 **SMBUS[3:0]**:

0: SMBus mode not implemented

1: SMBus mode implemented

50.9.13 I2C version register (I2C_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

50.9.14 I2C identification register (I2C_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Identifier.

50.9.15 I2C size identification register (I2C_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier.

50.9.16 I2C register map

The table below provides the I2C register map and the reset values.

Table 362. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_ TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_ TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x030- 0x03EC	Reserved																																

Table 362. I2C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03F0	I2C_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.				WKP[3:0]				ASYN[3:0]				SMBUS[3:0]			
	Reset value																					0	0	0	1	0	0	0	1	0	0	0	1
0x03F4	I2C_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0x03F8	I2C_IPIDR	ID[31:16]																ID[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0x03FC	I2C_SIDR	SID[31:16]																SID[15:0]															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

51 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART).

51.1 USART introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

51.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

51.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

51.4 USART implementation

The table(s) below describe(s) USART implementation. It (they) also include(s) LPUART for comparison.

Table 363. USART features

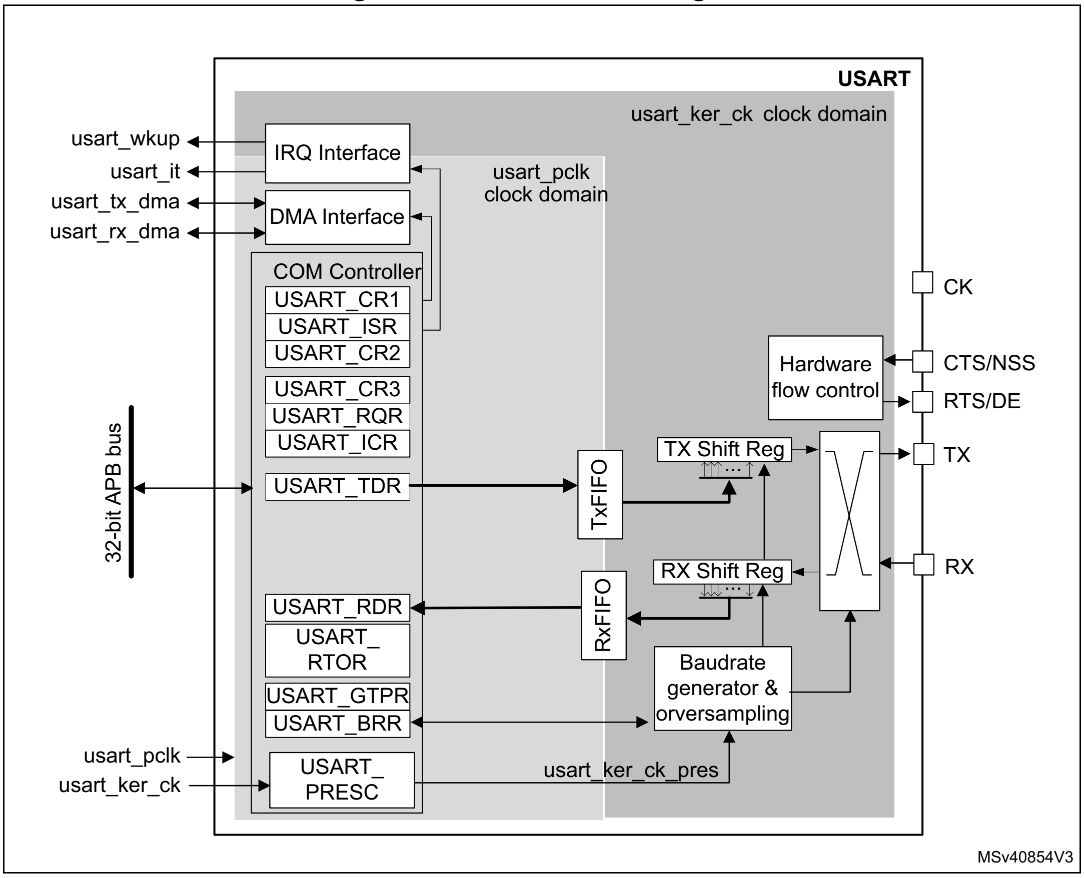
USART modes/features(1)	USART1/2/3/6	UART4/5/7/8
Hardware flow control for modem	X	X
Continuous communication using DMA	X	X
Multiprocessor communication	X	X
Synchronous mode (Master/Slave)	X	-
Smartcard mode	X	-
Single-wire half-duplex communication	X	X
IrDA SIR ENDEC block	X	X
LIN mode	X	X
Dual clock domain and wake-up from low-power mode	X	X
Receiver timeout interrupt	X	X
Modbus communication	X	X
Auto baud rate detection	X	X
Driver Enable	X	X
USART data length	7, 8 and 9 bits	
Tx/Rx FIFO	X	X
Tx/Rx FIFO size	16	
Wake-up from low-power mode	X(2)	X(2)

1. X = supported.
2. Wake-up from Stop, LP-Stop, and LPLV-Stop modes.

51.5 USART functional description

51.5.1 USART block diagram

Figure 544. USART block diagram



The simplified block diagram given in *Figure 544* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is disabled, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

51.5.2 USART signals

USART bidirectional communications

USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

- **RX** (Receive Data Input)

  RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.
- **TX** (Transmit Data Output)

  When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

RS232 hardware flow control mode

The following pins are required in RS232 hardware flow control mode:

- **CTS** (Clear To Send)

  When driven high, this signal blocks the data transmission at the end of the current transfer.
- **RTS** (Request To Send)

  When it is low, this signal indicates that the USART is ready to receive data.

RS485 hardware flow control mode

The following pin is required in RS485 hardware control mode:

- **DE** (Driver Enable)

  This signal activates the transmission mode of the external transceiver.

Synchronous SPI master/slave mode and smartcard mode

The following pin is required in synchronous master/slave mode and smartcard mode:

- **CK**

  This pin acts as clock output in synchronous SPI master and smartcard modes.

  It acts as clock input in synchronous SPI slave mode.

  In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable.

  In smartcard mode, CK output provides the clock to the smartcard.
- **NSS**

  This pin acts as slave Select input in synchronous slave mode.

Refer to *Table 364* and *Table 365* for the list of USART input/output pins and internal signals.

Table 364. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS(3)/UART_CTS	Input	Clear to send
USART_RTS(1)/UART_RTS(2)	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK(1)	Input/Output	Clock input in synchronous slave mode. Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE/USART_RTS share the same pin.
2. UART_DE/UART_RTS share the same pin.
3. USART_CTS and USART_NSS share the same pin.

Description of USART input/output signals

Table 365. USART internal input/output signals

Pin name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

51.5.3 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 545*):

- 7-bit character length: M[1:0] = ‘10’
- 8-bit character length: M[1:0] = ‘00’
- 9-bit character length: M[1:0] = ‘01’

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

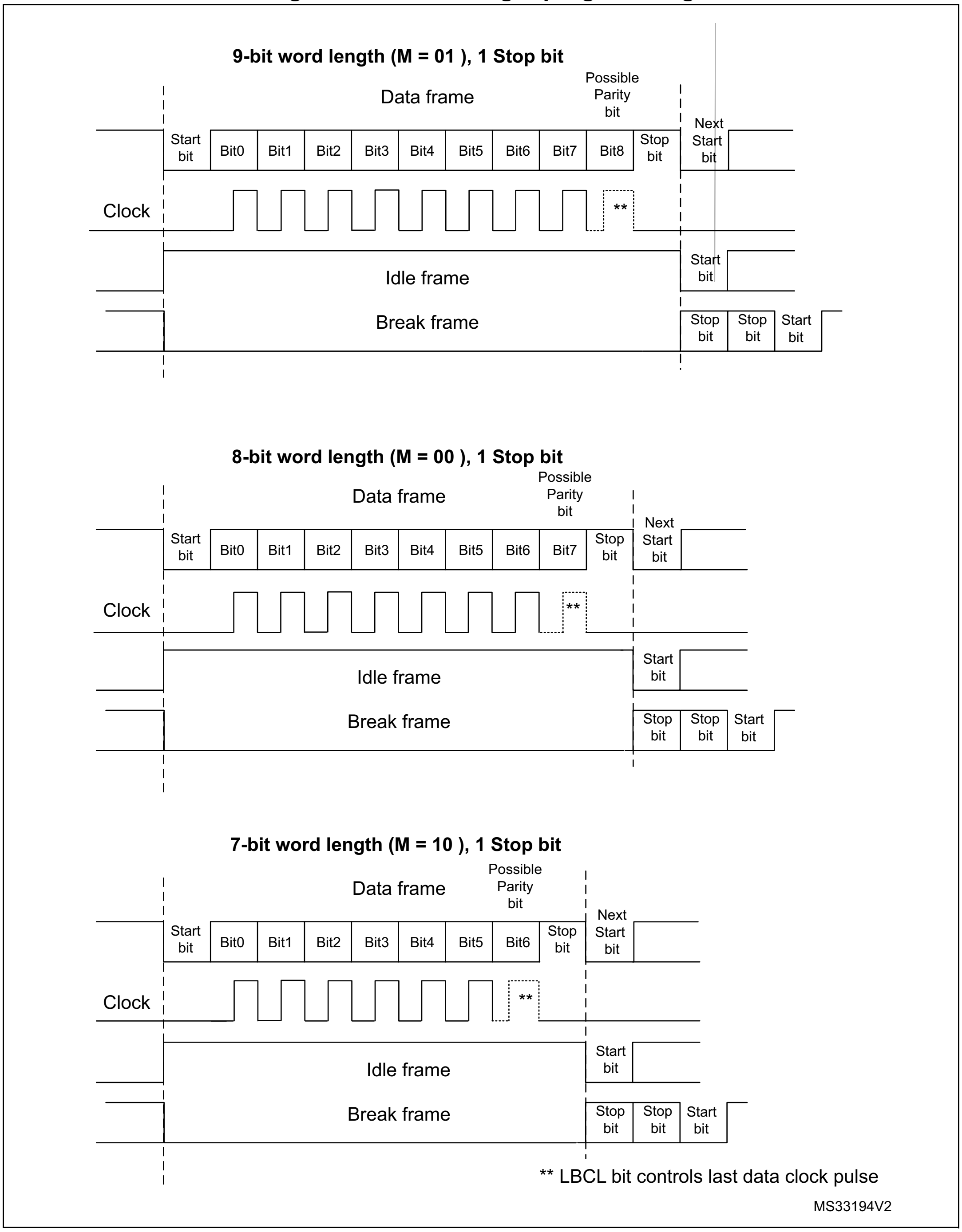
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 545. Word length programming



51.5.4 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The RXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated, when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bits fields.

  This means that the RXFIFO is filled until the number of data in the RXFIFO is equal to the programmed threshold.

  RXFTCFG data have been received: one data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to ‘101’, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data is not set the overrun flag.
- The TXFT flag is set in the USART_ISR register and the corresponding interrupt (if enabled) is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bits fields.

  This means that the TXFIFO is emptied until the number of empty locations in the TXFIFO is equal to the programmed threshold.

51.5.5 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit should not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

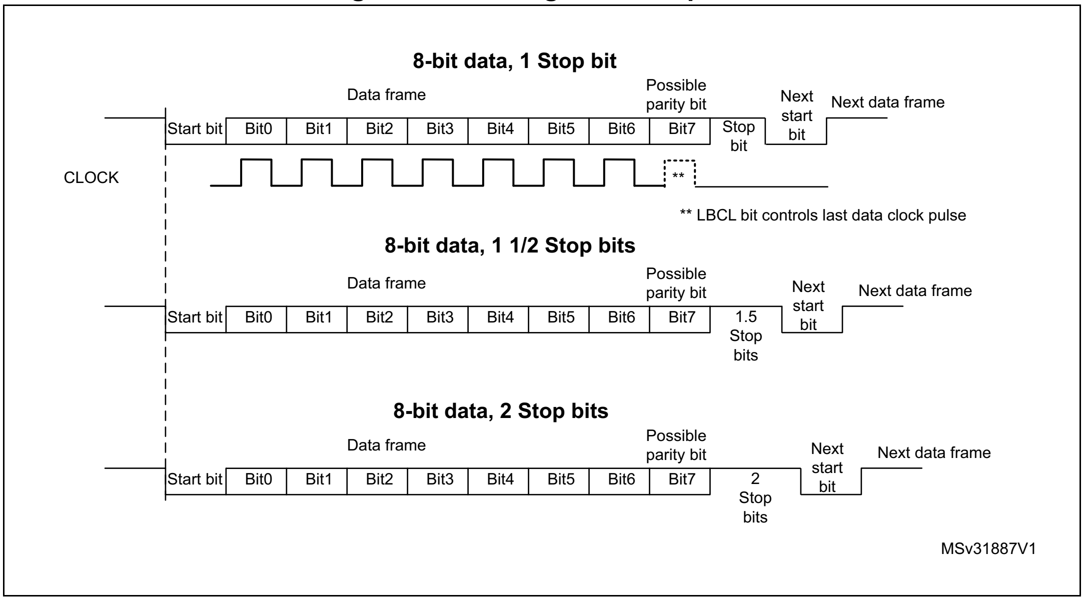
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission features 10 low bits (when M[1:0] = '00') or 11 low bits (when M[1:0] = '01') or 9 low bits (when M[1:0] = '10') followed by 2 stop bits (see *Figure 546*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 546. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 51.5.19: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame is complete.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  - the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  - the TXFIFO is not full;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

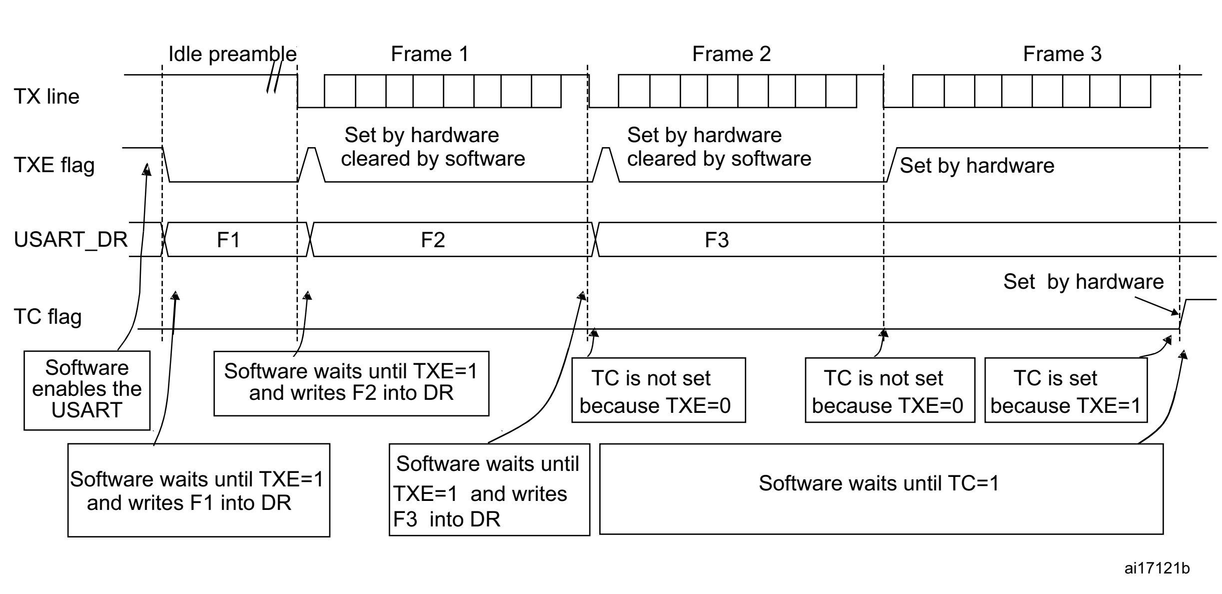
  When the TXFIFO is not full, the TXFNF flag stays at ‘1’ even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the device to enter the low-power mode (see *Figure 547: TC/TXE behavior when transmitting*).

Figure 547. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 545*).

If a '1' is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is completed (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

51.5.6 USART receiver

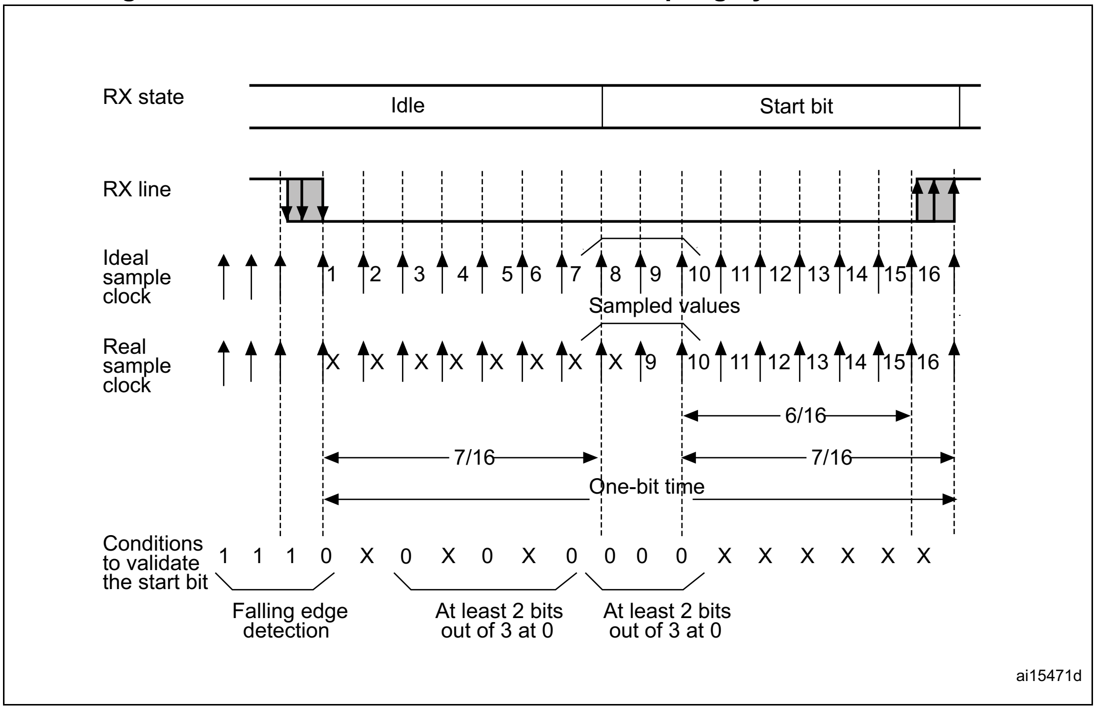
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 548. Start bit detection when oversampling by 16 or 8



Note: If the sequence is not complete, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at ‘0’ (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at ‘0’ and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at ‘0’).

The start bit is validated but the NE noise flag is set if,

a) for both samplings, 2 out of the 3 sampled bits are at ‘0’ (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits)

or

b) for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at ‘0’.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to ‘1’.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 51.5.19: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  – When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  – When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty i.e. when there are data to be read from the RXFIFO.
- In single buffer mode:
  – When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to ‘1’ in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  – When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to ‘1’ in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.
- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  - The ORE bit is set.
  - The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section *Reset and clock control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

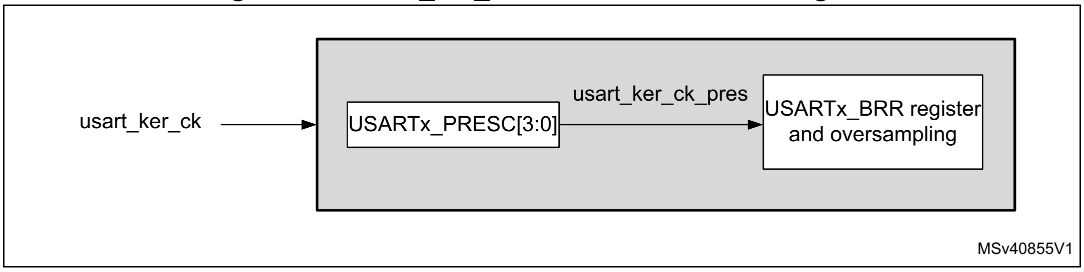
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see Section *Reset and clock control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 549. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 550* and *Figure 551*).

Depending on your application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 51.5.8: Tolerance of the USART receiver to clock deviation on page 2471*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum

usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on your application:
  - select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Figure 366*) because this indicates that a glitch occurred during the sampling.
  - select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 51.5.8: Tolerance of the USART receiver to clock deviation on page 2471*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NECF bit in USART_ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to '0 ' by hardware.

Figure 550. Data sampling when oversampling by 16

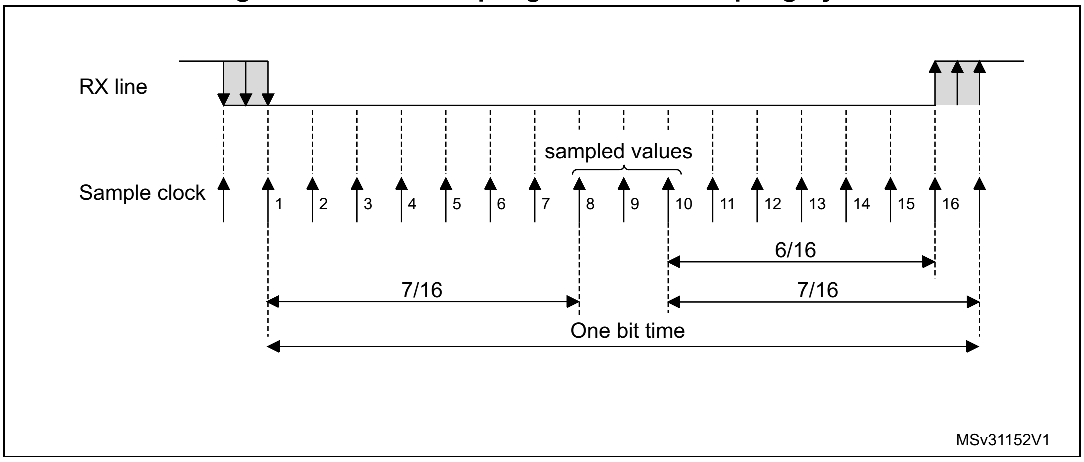


Figure 551. Data sampling when oversampling by 8

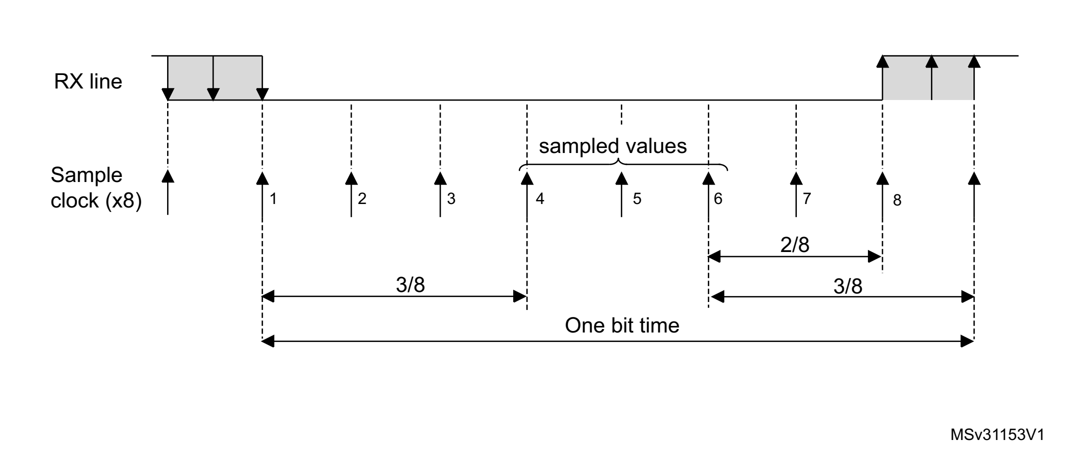


Table 366. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing ‘1’ to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- 1.5 stop bits (smartcard mode)

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 51.5.16: USART receiver timeout on page 2485* for more details).
- 2 stop bits

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

51.5.7 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = ‘0’ or ‘1’)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart\_ker\_ck\_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value should not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 baud with usart_ker_ck_pres = 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >> 1 = 0x1
  BRR = 0x681

Example 2

To obtain 921.6 Kbaud with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR = USARTDIV = 0d52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR = 0x64

51.5.8 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 367*, *Table 368*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 367. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 368. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 367 and Table 368 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

51.5.9 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at ‘1’.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a ‘0’).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection should be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

51.5.10 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to ‘1’. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

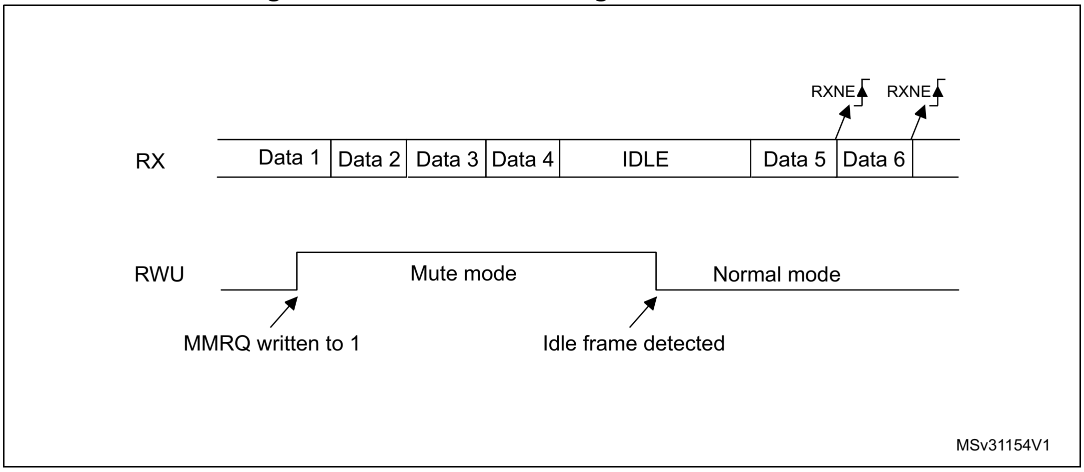
- Idle Line detection if the WAKE bit is reset,
- Address Mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to ‘1’ and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 552*.

Figure 552. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a ‘1’, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software should ensure that there is at least one empty location in the RXFIFO before entering mute mode.

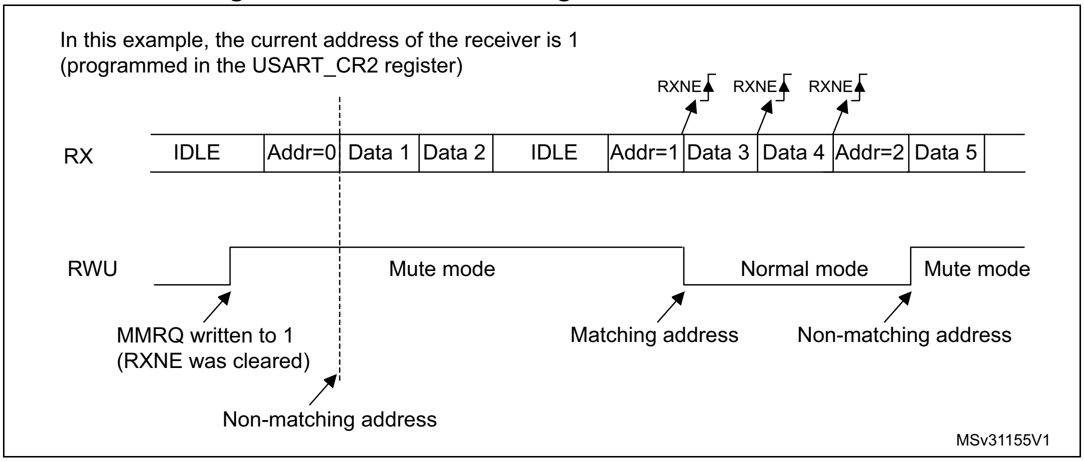
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 553*.

Figure 553. Mute mode using address mark detection



51.5.11 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception is completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

51.5.12 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 369*.

Table 369. USART frame formats

M bits	PCE bit	USART frame[(1)]
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of "1s" inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits are set, the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of "1s" inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of "1s" if even parity is selected (PS = 0) or an odd number of "1s" if odd parity is selected (PS=1).

51.5.13 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 51.4: USART implementation on page 2454*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 51.5.4* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 ‘0 bits as a break character. Then two bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as ‘0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a ‘1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (i.e. stop bit detected at ‘0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the *Figure 554*.

Examples of break frames are given on *Figure 555*.

Figure 554. Break detection in LIN mode (11-bit break length - LBDL bit is set)

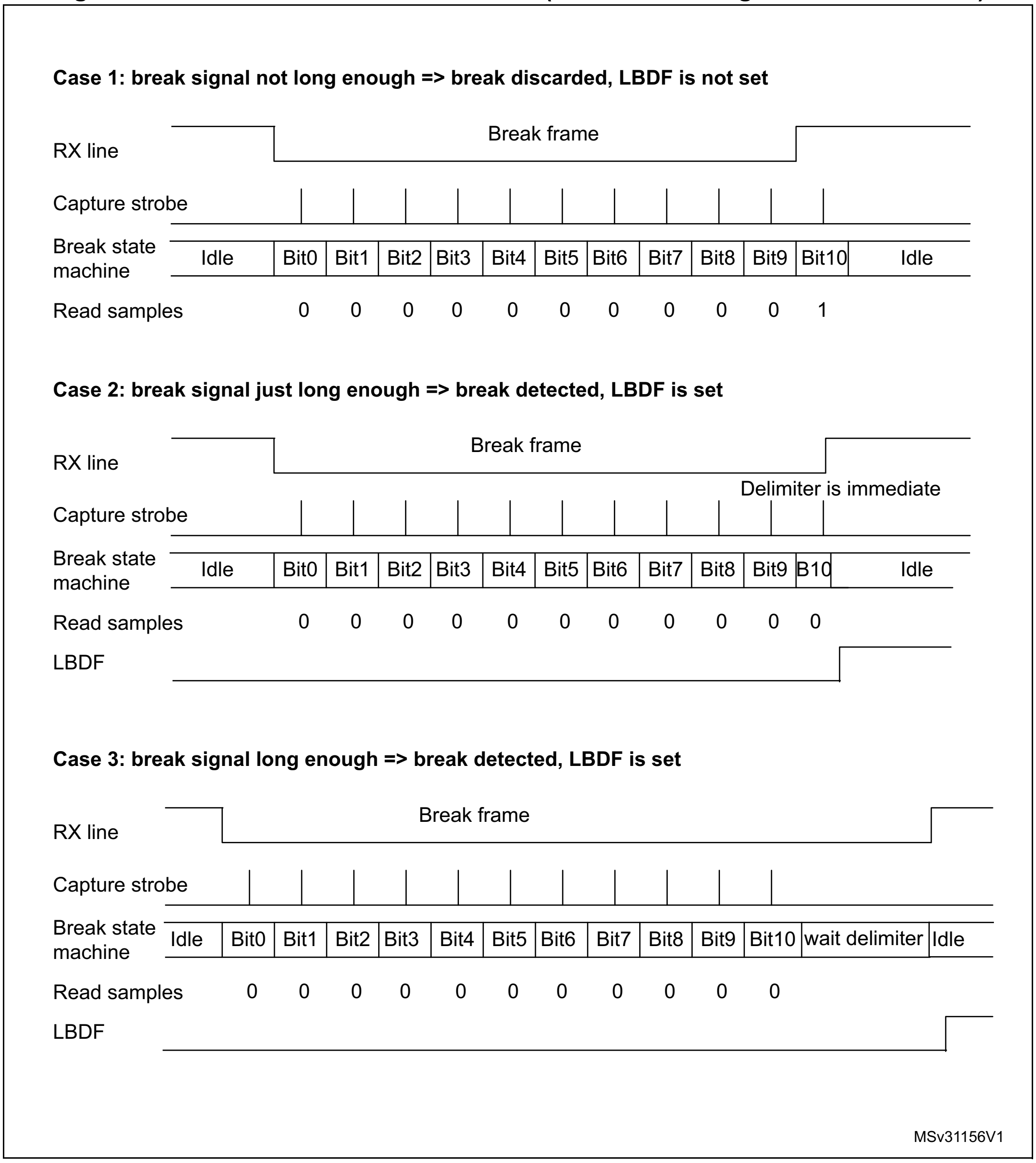
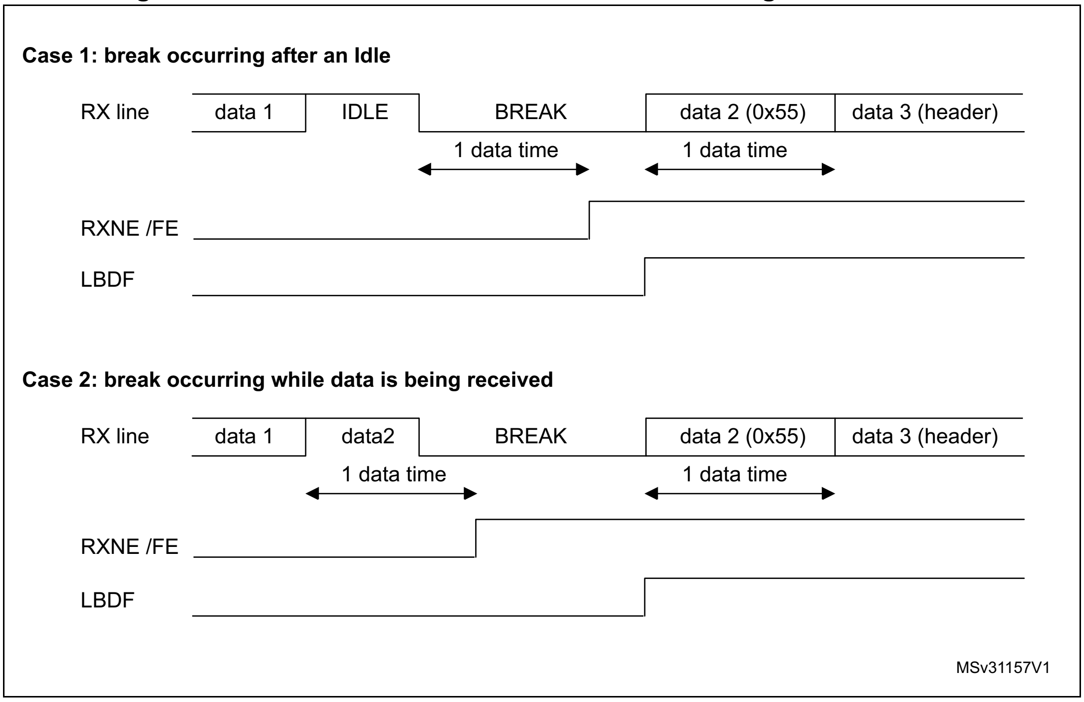


Figure 555. Break detection in LIN mode vs. Framing error detection



51.5.14 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to '1'. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 556*, *Figure 557* and *Figure 558*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 556. USART example of synchronous master transmission

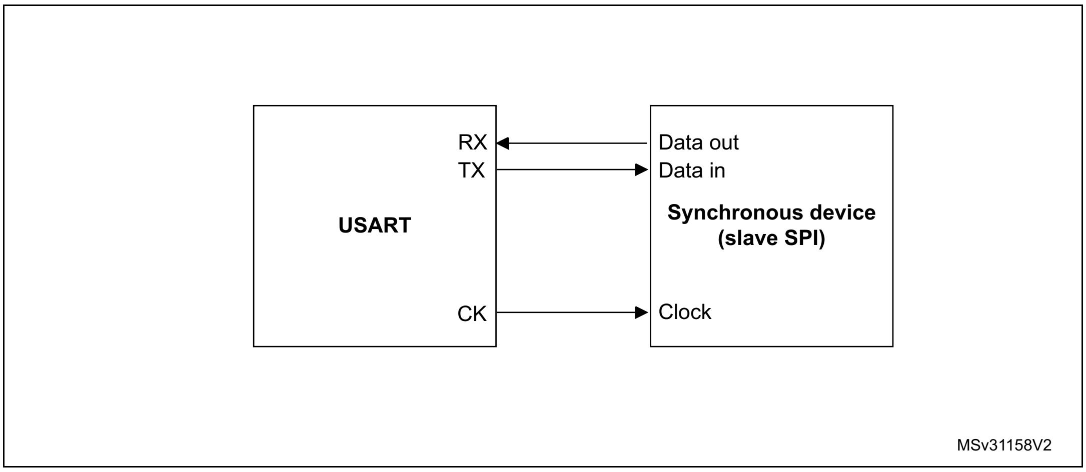


Figure 557. USART data clock timing diagram in synchronous master mode (M bits = 00)

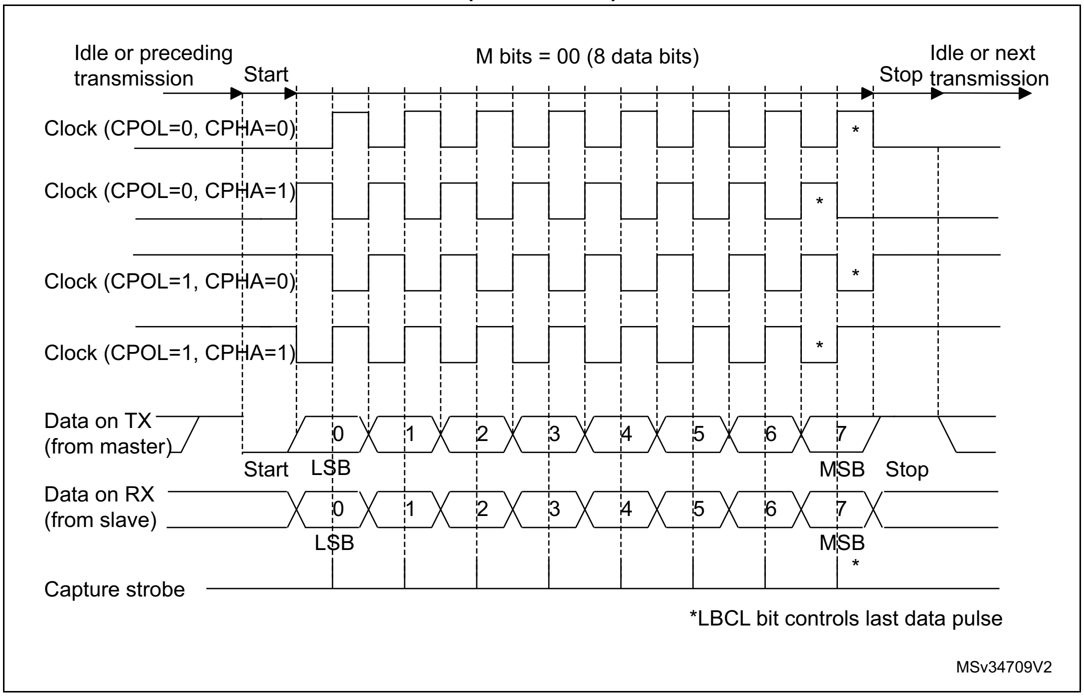
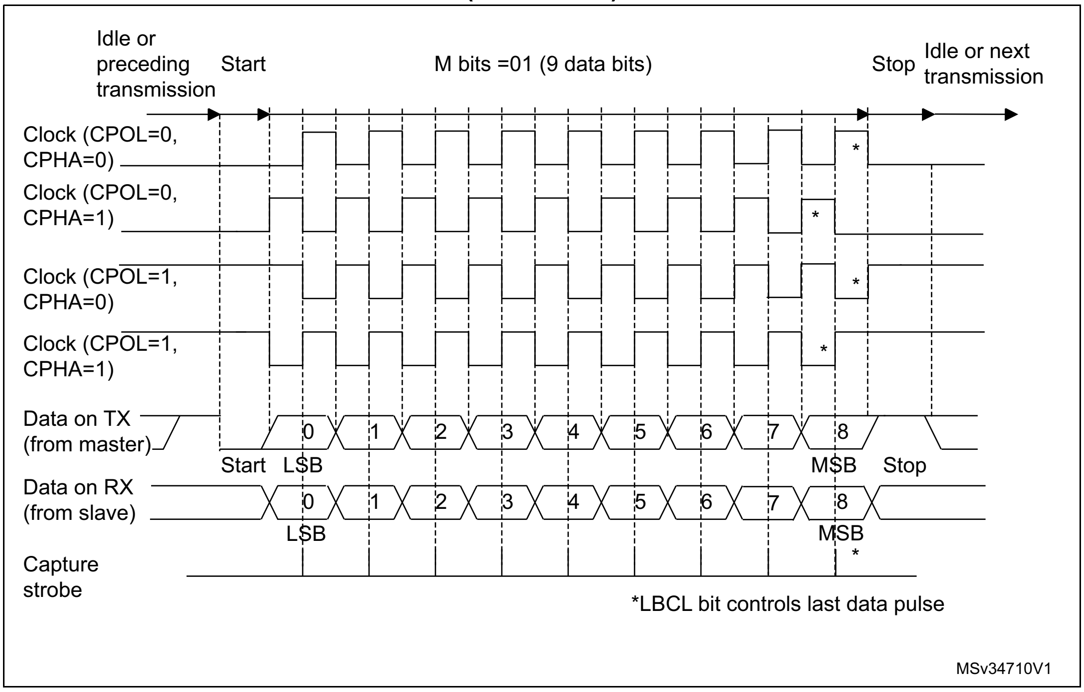


Figure 558. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to '1'. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

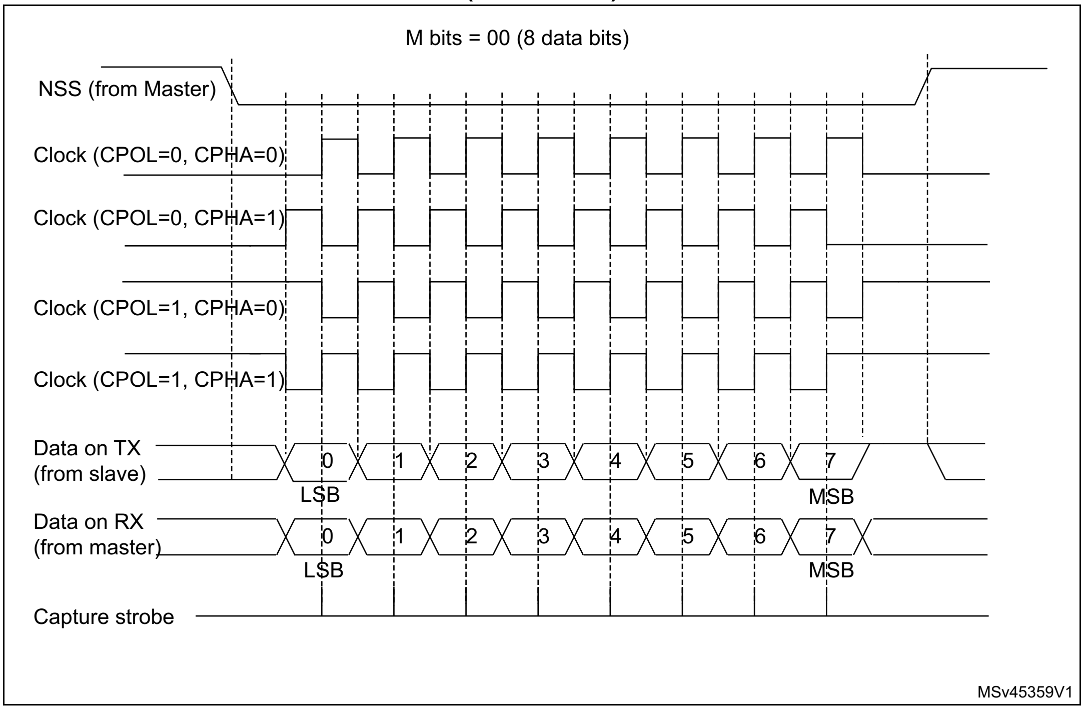
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 559*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 559. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  The SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR should be written 3 usart_ker_ck *cycles before the first CK edge.*

51.5.15 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to ‘1’:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

51.5.16 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = ‘00’ or STOP = ‘11’
- from the end of the second stop bit if STOP = ‘10’.
- from the beginning of the stop bit if STOP = ‘01’.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

51.5.17 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 51.4: USART implementation on page 2454*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

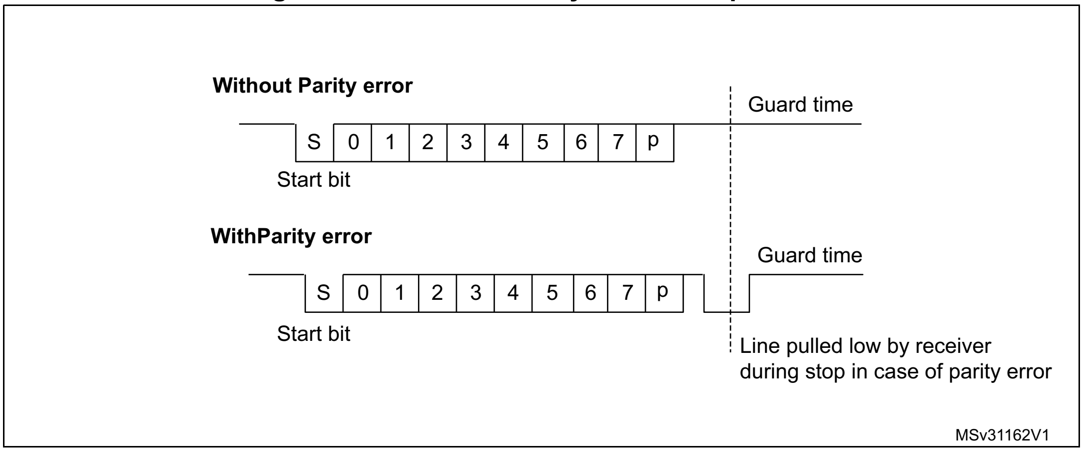
The USART should be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = '11' in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 560 shows examples of what can be seen on the data line with and without parity error.

Figure 560. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

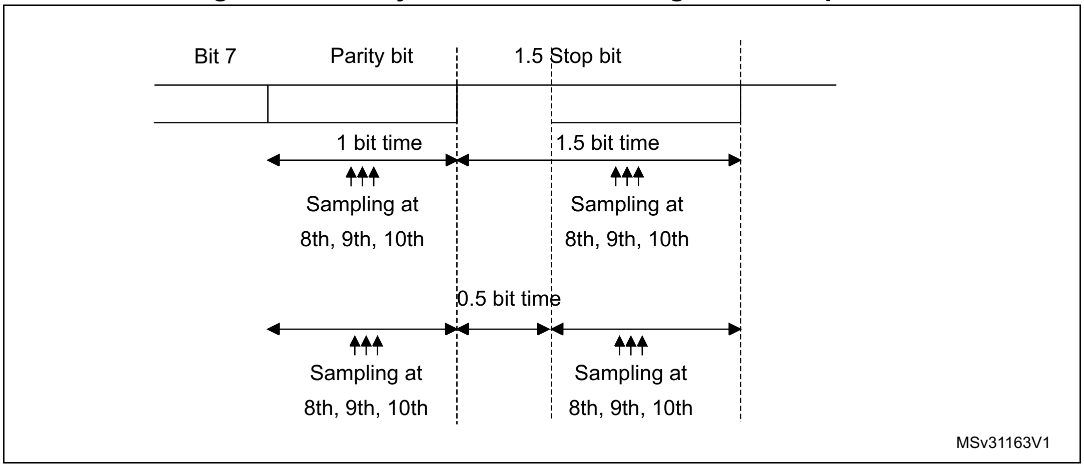
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.
- In transmission, the USART inserts the Guard Time (as programmed in the Guard Time register) between two successive characters. As the Guard Time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT (Character Guard Time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the Guard Time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the Guard Time counter to count up to the programmed value in the Guard Time register. TC is forced low during this time. When the Guard Time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The deassertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 561 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 561. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: *The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.*

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: *As in the smartcard protocol definition, the BWT/CWT values should be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT -11, respectively, taking into account the length of the last character itself.*

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN = LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN = LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV = 1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, assuming that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH results in a USART received character of 03 and an odd parity.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART.

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

51.5.18 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 51.4: USART implementation on page 2454*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 562*).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbaud for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission should be avoided as the data to be transmitted could be corrupted.

- A ‘0‘ is transmitted as a high pulse and a ‘1’ is transmitted as a ‘0’. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 563*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder doesn’t work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART should discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power Baud clock (PSC value in the USART_GTPR).

Note: A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).

Figure 562. IrDA SIR ENDEC block diagram

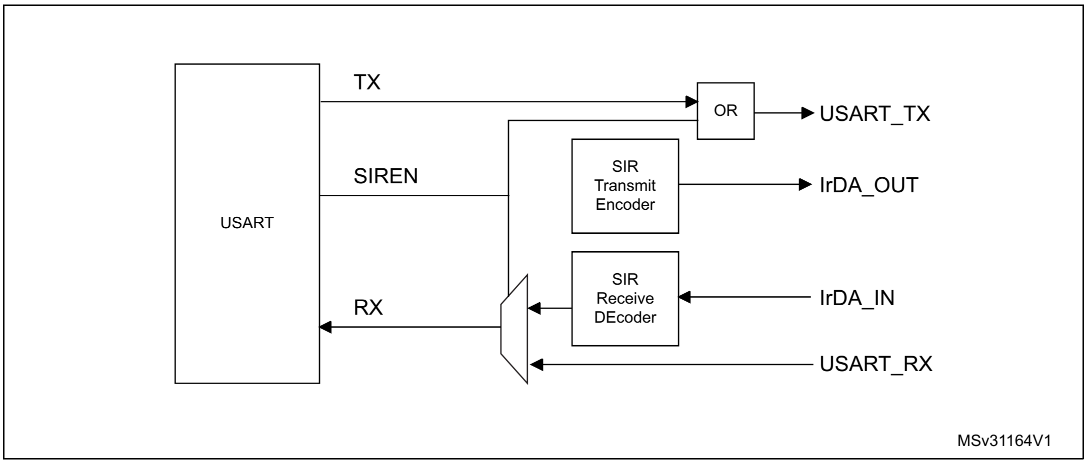
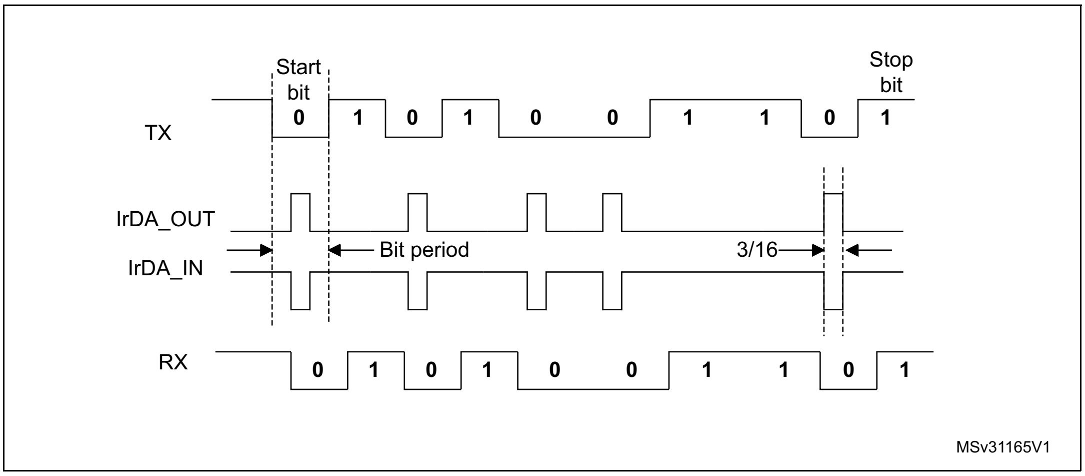


Figure 563. IrDA data modulation (3/16) - normal mode



51.5.19 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 51.4: USART implementation on page 2454 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 51.5.6. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

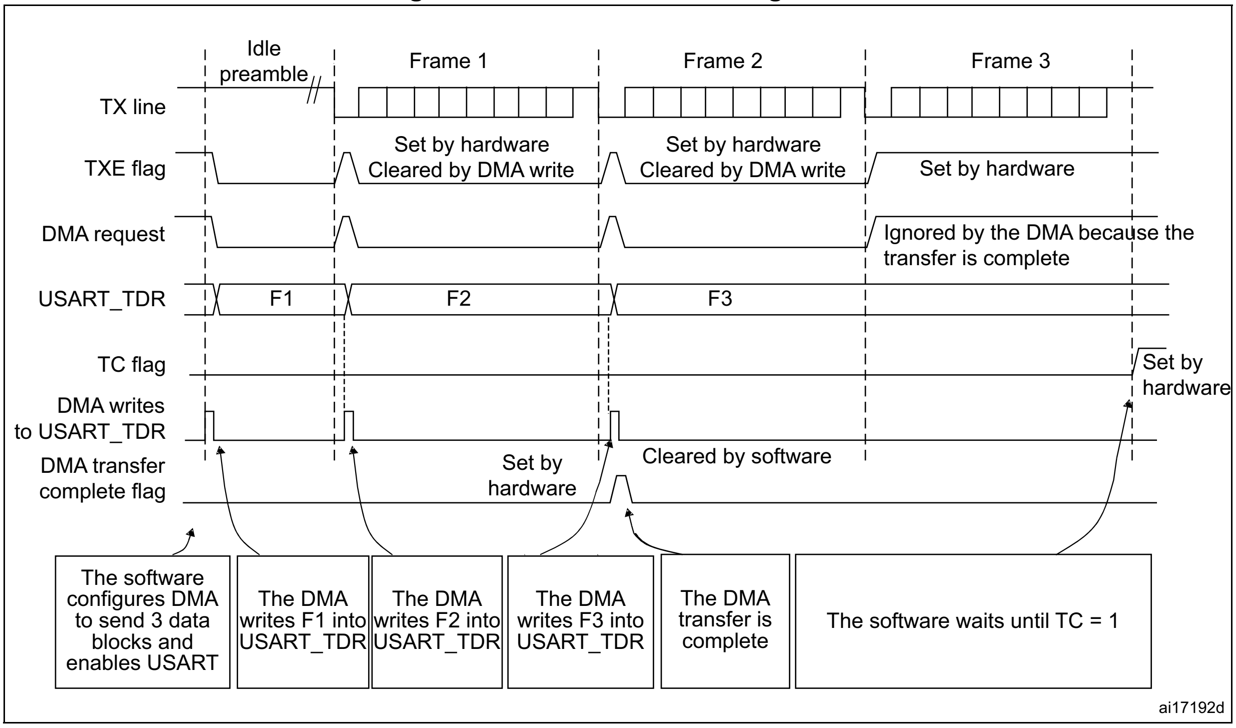
DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication is complete. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Figure 564. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (i.e. TXFNF = 1).

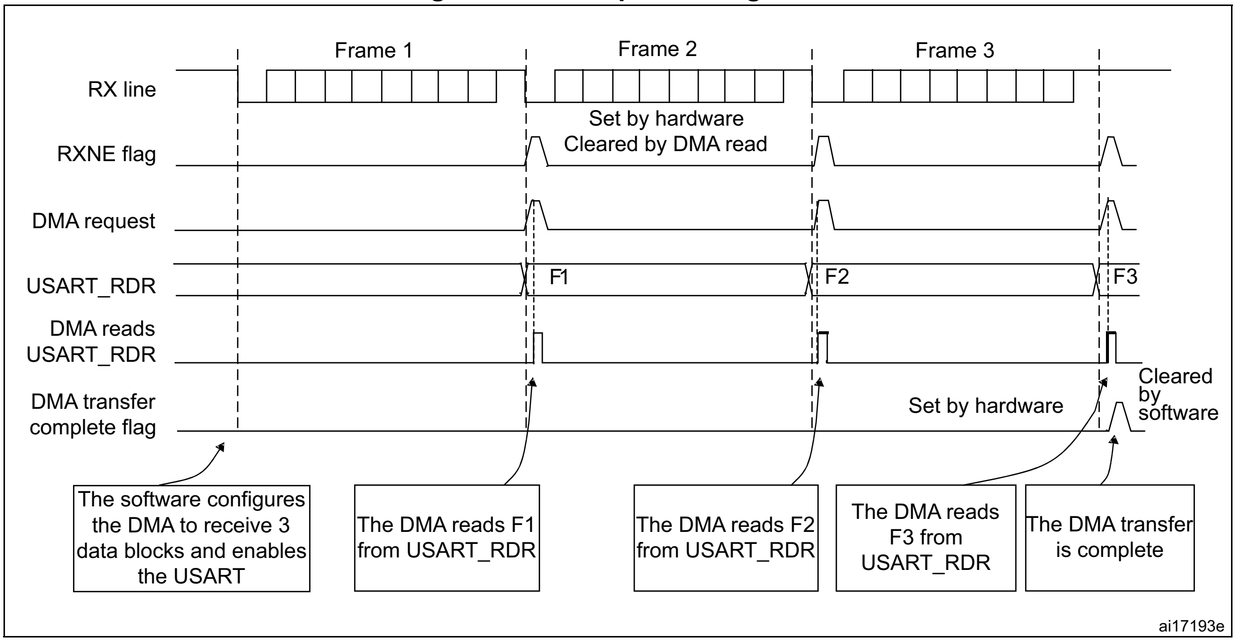
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to the corresponding *Direct memory access controller* section) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 565. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (i.e. RXFNE = 1).

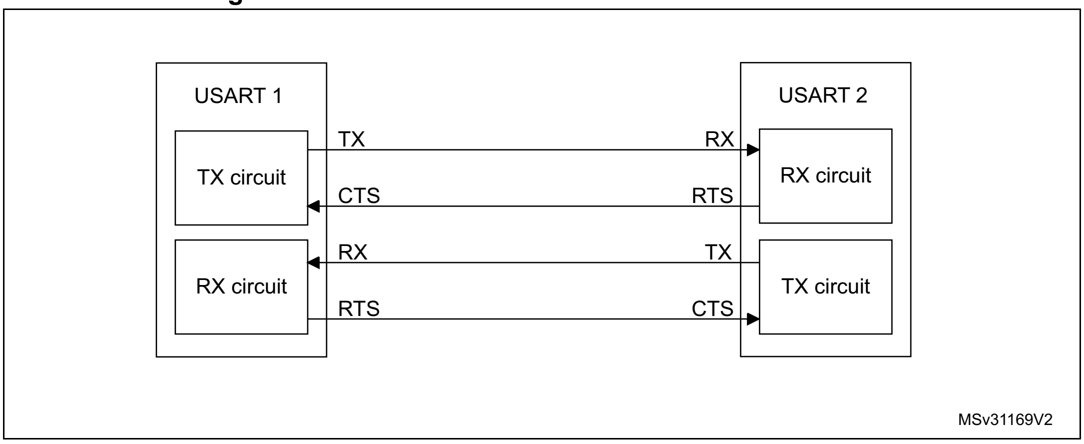
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

51.5.20 RS232 hardware flow control and RS485 Driver Enable

It is possible to control the serial data flow between 2 devices by using the CTS input and the RTS output. The *Figure 566* shows how to connect 2 devices in this mode:

Figure 566. Hardware flow control between 2 USARTs

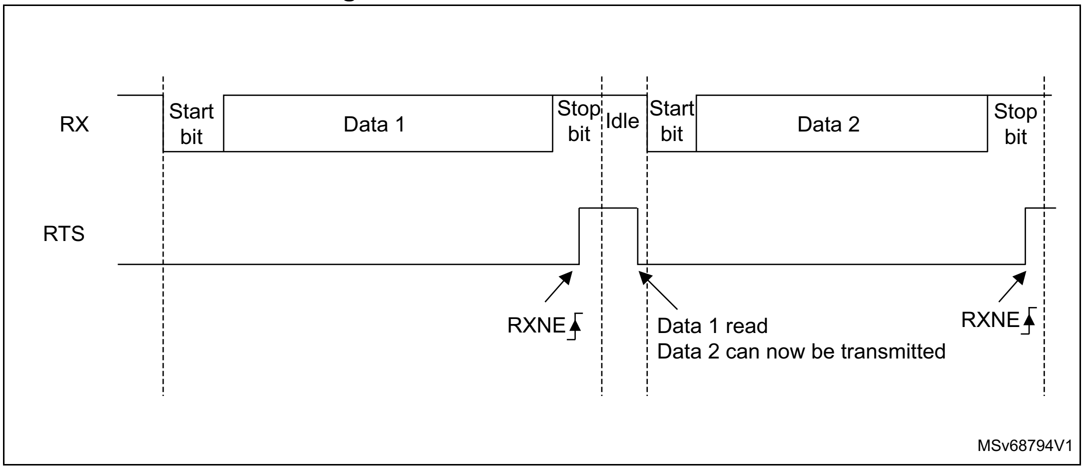


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to ‘1’ in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 567* shows an example of communication with RTS flow control enabled.

Figure 567. RS232 RTS flow control



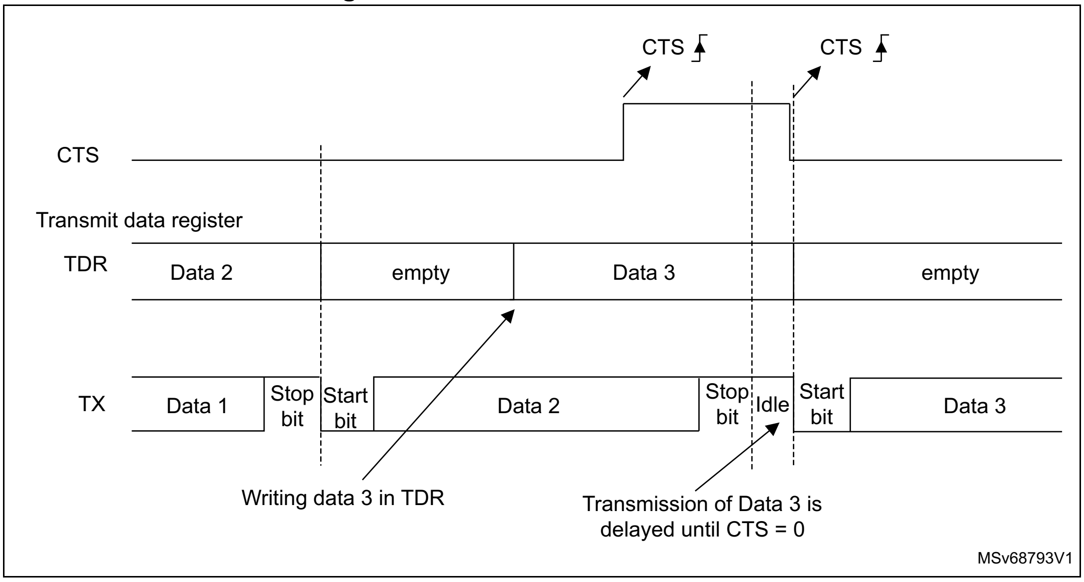
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission is completed before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 568* shows an example of communication with CTS flow control enabled.

Figure 568. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it should be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

51.5.21 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to ‘1’. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (i.e.the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 569 shows the USART behavior when the wake-up event is verified.

Figure 569. Wake-up event verified (wake-up event = address match, FIFO disabled)

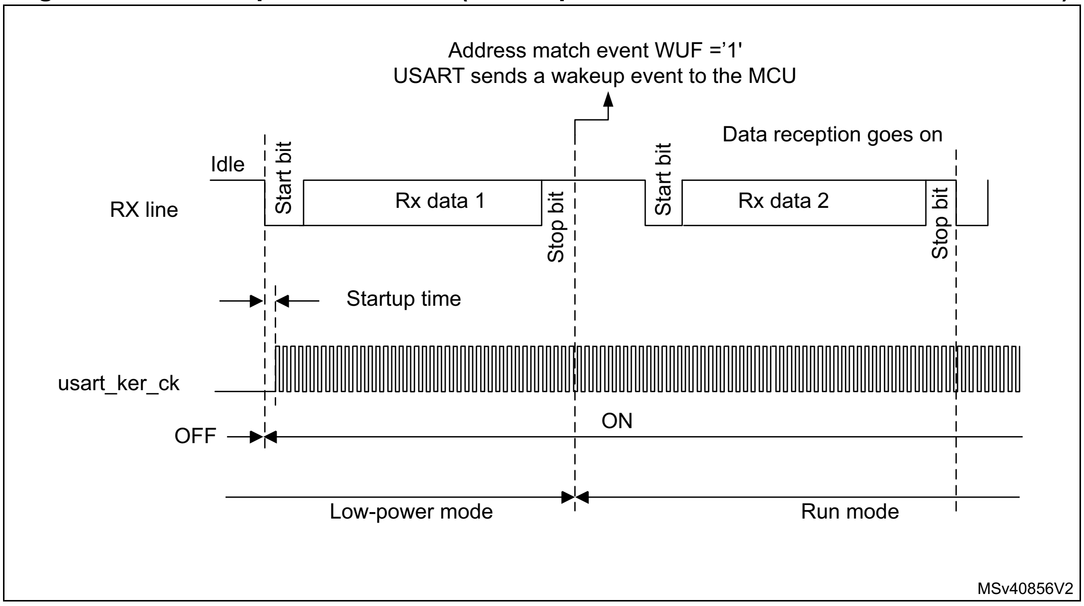
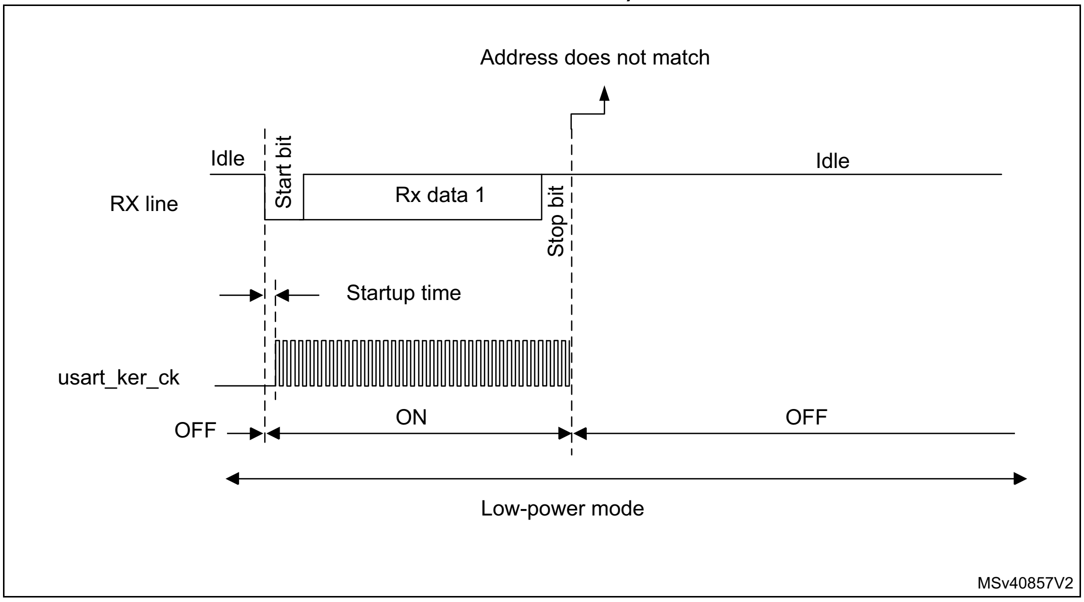


Figure 570 shows the USART behavior when the wake-up event is not verified.

Figure 570. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the device from low-power mode

The maximum baud rate that enables to correctly wake up the device from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 51.5.8: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = ‘01’, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 367: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the usart_ker_ck inaccuracy.

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% - 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 Kbaud.

51.6 USART in low-power modes

Table 370. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 51.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

51.7 USART interrupts

Refer to *Table 371* for a detailed description of all USART interrupt requests.

Table 371. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
	Transmit data register empty	TXE	TXEIE	Write TDR		No	
	Transmit FIFO not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
USART or UART	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR	Yes	Yes	No
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission Complete Before Guard Time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ		Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RXNEIE/ RXFNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
USART or UART	LIN break	LBDF	LBDIE	Write 1 in LBDCF	Yes	No	No
	Noise error in multibuffer communication	NE		Write 1 in NFCF		No	
	Overrun error in multibuffer communication	ORE(3)	EIE	Write 1 in ORECF		No	
	Framing Error in multibuffer communication	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 51.4: USART implementation* for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

51.8 USART registers

Refer to Section 1.2 on page 115 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

51.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit
M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit
M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End-of-block interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 51.4: USART implementation on page 2454.

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16
1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not-full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit (‘0’ followed by ‘1’) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to ‘1’. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

51.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = '00': 1 start bit, 8 Data bits, n Stop bit

M[1:0] = '01': 1 start bit, 9 Data bits, n Stop bit

M[1:0] = '10': 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of Bbock interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 51.4: USART implementation on page 2454.

Bits 25:21 **DEAT[4:0]**: Driver enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 20:16 **DEDT[4:0]**: Driver enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to *Section 51.4: USART implementation on page 2454**.*

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: During transmission, a low pulse on the TE bit ('0' followed by '1') sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1'. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode and clear it when exit from low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK pin is always available when CLKEN = 1, regardless of the UE bit value.

51.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 and Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H). The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd = 0/mark)

1: TX pin signal values are inverted (V_{DD} =0/mark, Gnd = 1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} =1/idle, Gnd = 0/mark)

1: RX pin signal values are inverted (V_{DD} =0/mark, Gnd = 1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:

UE = 0

SCEN = 1

GTPR configuration

CLKEN= 1

UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 550* and *Figure 551*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**: NSS pin enable

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.

1: Slave mode enabled.

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Note: The CPOL, CPHA and LBCL bits should not be written while the transmitter is enabled.

51.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			TCBGTIE	TXFTIE	WUFIE	WUS[1:0]		SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration

000:TXFIFO reaches 1/8 of its depth

001:TXFIFO reaches 1/4 of its depth

010:TXFIFO reaches 1/2 of its depth

011:TXFIFO reaches 3/4 of its depth

100:TXFIFO reaches 7/8 of its depth

101:TXFIFO becomes empty

Remaining combinations: Reserved

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration

000:Receive FIFO reaches 1/8 of its depth

001:Receive FIFO reaches 1/4 of its depth

010:Receive FIFO reaches 1/2 of its depth

011:Receive FIFO reaches 3/4 of its depth

100:Receive FIFO reaches 7/8 of its depth

101:Receive FIFO becomes full

Remaining combinations: Reserved

Bit 24 **TCBGTIE**: Transmission complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmit mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 51.4: USART implementation on page 2454.

Bit 13 **DDRE**: DMA Disable on reception error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred (used for smartcard mode).

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data is written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables checking the communication flow w/o reading the data

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission is completed before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half duplex mode is not selected

1: Half duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

51.8.5 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]**: USART baud rate

BRR[15:4]

BRR[15:4] = USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

51.8.6 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The Transmission Complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 7:0 **PSC[7:0]**: Prescaler value

In IrDA low-power and normal IrDA mode:

PSC[7:0] = IrDA normal and low-power baud rate

PSC[7:0] is used to program the prescaler for dividing the USART source clock to achieve the low-power frequency: the source clock is divided by the value given in the register (8 significant bits):

In smartcard mode:

PSC[4:0] = Prescaler value

PSC[4:0] is used to program the prescaler for dividing the USART source clock to provide the smartcard clock. The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value
00001: Divides the source clock by 1 (IrDA mode) / by 2 (smartcard mode)
00010: Divides the source clock by 2 (IrDA mode) / by 4 (smartcard mode)
00011: Divides the source clock by 3 (IrDA mode) / by 6 (smartcard mode)
...
11111: Divides the source clock by 31 (IrDA mode) / by 62 (smartcard mode)
0010 0000: Divides the source clock by 32 (IrDA mode)
...
1111 1111: Divides the source clock by 255 (IrDA mode)

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.
This bitfield is reserved and forced by hardware to '0' when the smartcard and IrDA modes are not supported. Refer to Section 51.4: USART implementation on page 2454.

51.8.7 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block length

This bitfield gives the Block length in smartcard T = 1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

BLEN = 0: 0 information characters + LEC

BLEN = 1: 0 information characters + CRC

BLEN = 255: 254 information characters + CRC (total 256 characters))

In smartcard mode, the Block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the Block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bits during which there is no activity on the RX line.

In standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the Receiver timeout feature is not supported. Refer to *Section 51.4: USART implementation on page 2454*.

51.8.8 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing ‘1’ to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO i.e. clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software should wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

51.8.9 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0X80 00C0

X = 2 if FIFO/smartcard mode is enabled

X = 0 if FIFO is enabled and smartcard mode is disabled

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register i.e. the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the USART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the USART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to ‘101’, RXFT flag is set if 16 data are available i.e. 15 data in the RXFIFO and 1 data in the USART_RDR. Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is ‘1’. Refer to Section 51.4: USART implementation on page 2454.

Bit 24 **RXFF**: RXFIFO full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.

0: RXFIFO not full.

1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO empty

This bit is set by hardware when TXFIFO is empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.

0: TXFIFO not empty.

1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454.*

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454.*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454**.*

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF should be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXFE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 51.5.8: Tolerance of the USART receiver to clock deviation on page 2471).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

51.8.10 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data is complete and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete or transmission is complete unsuccessfully (i.e. a NACK is received from the card)

1: Transmission is complete successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is '1'. Refer to *Section 51.4: USART implementation on page 2454**.*

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454**.*

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454**.*

Bit 19 **RWU**: *Receiver wake-up from mute mode*

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454**.*

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: Break character transmitted

1: Break character requested by setting SBKRQ bit in USART_RQR register

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character Match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception on going

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation was completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454.*

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if the EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to the EOBCF in the USART_ICR register.

0: End of Block not reached

1: End of Block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to *Section 51.4: USART implementation on page 2454.*

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register not full

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data is complete and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is is cleared by software, by writing 1 to the TCCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission is not complete

1: Transmission is complete

Note: If TE bit is reset and no transmission is on going, the TC bit is set immediately.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (i.e. a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 or EIE = 1 in the LPUART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NECF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 51.5.8: Tolerance of the USART receiver to clock deviation on page 2471).

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in receiver mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

51.8.11 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 11 **RTOCF**: Receiver timeout clear flag

Writing 1 to this bit clears the RTOF flag in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 8 **LBDCF**: LIN break detection clear flag

Writing 1 to this bit clears the LBDF flag in the USART_ISR register.

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag

Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag

Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag

Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag

Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag

Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag

Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag

Writing 1 to this bit clears the PE flag in the USART_ISR register.

51.8.12 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Figure 544*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

51.8.13 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value
Contains the data character to be transmitted.
The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Figure 544*).
When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

51.8.14 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Remaining combinations: Reserved

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is 1011 i.e. input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 51.4: USART implementation on page 2454.

51.8.15 USART hardware configuration register 2 (USART_HWCFGR2)

Address offset: 0x3EC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[3:0]				CFG1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CFG2[3:0]**: USART hardware configuration 2
CFG2[3:0] = SPI_SLAVE

Bits 3:0 **CFG1[3:0]**: USART hardware configuration 1
CFG1[3:0] = LOG_FIFO_DEPTH_RX
RXFIFO size = $2^{LOG\_FIFO\_DEPTH\_RX}$

51.8.16 USART hardware configuration register 1 (USART_HWCFGR1)

Address offset: 0x3F0

Reset value: 0x4011 X11X

X = 1 for USARTx, X = 0 for UARTx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG8[3:0]				CFG7[3:0]				CFG6[3:0]				CFG5[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CFG8[3:0]**: USART hardware configuration 8
CFG8[3:0] = LOG_FIFO_DEPTH_TX
TXFIFO size = $2^{LOG\_FIFO\_DEPTH\_TX}$

Bits 27:24 **CFG7[3:0]**: USART hardware configuration 7
CFG7[3:0] = BAUDGEN_TYPE

Bits 23:20 **CFG6[3:0]**: USART hardware configuration 6
CFG6[3:0] = DUAL_clock

Bits 19:16 **CFG5[3:0]**: USART hardware configuration 5
CFG5[3:0] = ABR_RTO_SCT1

Bits 15:12 **CFG4[3:0]**: USART hardware configuration 4
CFG4[3:0] = SMARTCARD_SEL

Bits 11:8 **CFG3[3:0]**: USART hardware configuration 3
CFG3[3:0] = IRDA_SEL

Bits 7:4 **CFG2[3:0]**: USART hardware configuration 2
CFG2[3:0] = LIN_MASTER

Bits 3:0 **CFG1[3:0]**: USART hardware configuration 1
CFG1[3:0] = SCLK_EXTRACT

51.8.17 USART version register (USART_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision
These bits return the USART major revision.

Bits 3:0 **MINREV[3:0]**: Minor revision
These bits return the USART minor revision.

51.8.18 USART Identification register (USART_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]:** Peripheral identifier
These bits return the USART identifier ID[31:0]:
ID[31:0] = 0x00130003

51.8.19 USART Size Identification register (USART_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]:** Size identification

These bits return the size of the memory region allocated to USART registers.

The size of this memory region is of 1 Kbytes.

SID[31:0] = 0xA3C5 DD01

51.8.20 USART register map

The table below gives the USART register map and reset values.

Table 372. USART register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			TCBGTIE	TXFTIE	WUFIE	WUS [1:0]		SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					X	X	X	X	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 372. USART register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0
0x3EC	USART_ HWCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFG2[3:0]				CFG1[3:0]			
	Reset value																									0	0	0	1	0	0	1	0
0x3F0	USART_ HWCFGR1	CFG8[3:0]				CFG7[3:0]				CFG6[3:0]				CFG5[3:0]				CFG4[3:0]				CFG3[3:0]				CFG2[3:0]				CFG1[3:0]			
	Reset value (USARTx)	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
	Reset value (UARTx)	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
0x3F4	USART_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
0x3F8	USART_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x3FC	USART_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

52 Serial peripheral interface (SPI)

52.1 Introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The (SPI) interface supports a half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and is capable of operating in multi slave or multi master configurations. In case of master configuration it provides the communication clock (SCK) to the external slave device. The slave select signal can be provided by the master and accepted by the slave optionally, too. The Motorola data format is used by default, but some other specific modes are supported as well.

52.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection[a]
- Multi master or multi slave mode capability
- Dual clock domain, separated clock for the peripheral kernel which can be independent of APB bus clock
- 8 master mode baud rate prescalers up to kernel frequency/2
- Protection of configuration and setting
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transaction to control SS and CRC
- Dedicated transmission and reception flags with interrupt capability
- Slave transmission and/or reception capability in Stop mode (no clock provided to the peripheral) with wake up
- SPI Motorola and TI formats support
- Hardware CRC feature can verify integrity of the communication at the end of transaction by:
  – Adding CRC value at Tx mode
  – Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault and the frame error at dependency on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instance)
- Configurable FIFO thresholds (data packing)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)

52.3 SPI implementation

Table 373. STM32MP153A SPI features

SPI modes/features	SPI2S1	SPI2S2	SPI2S3	SPI4	SPI5	SPI6
Rx & Tx FIFO size [x 8-bit]	16	16	16	8	8	8

a. 24- and 32-bit data width are not always available. Refer to *Section 52.3: SPI implementation*.

Table 373. STM32MP153A SPI features (continued)

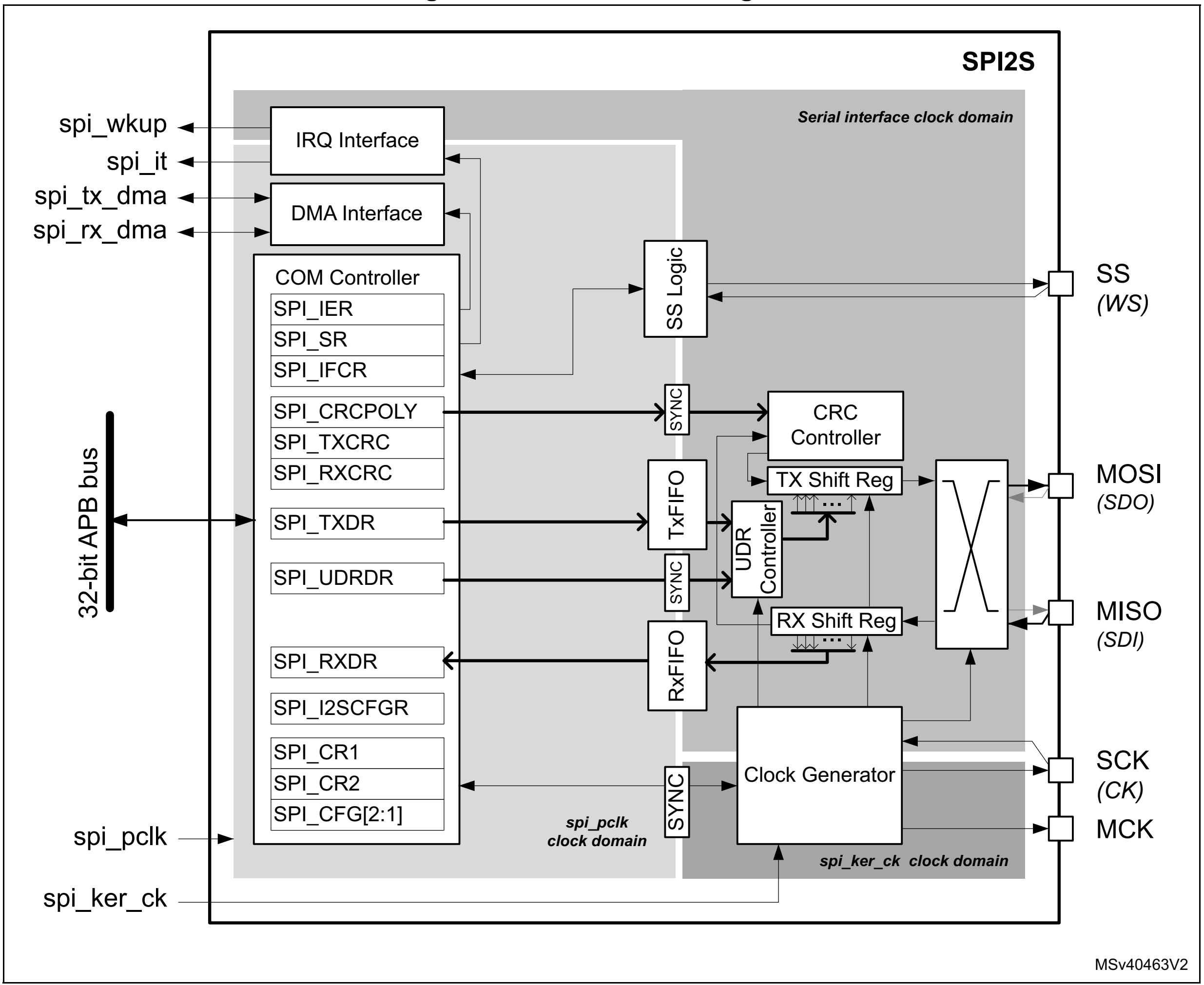
SPI modes/features	SPI2S1	SPI2S2	SPI2S3	SPI4	SPI5	SPI6
Maximum configurable data and CRC size [bits]	32	32	32	16	16	16
I2S feature	Yes	Yes	Yes	No	No	No

52.4 SPI functional description

52.4.1 SPI block diagram

The SPI allows a synchronous, serial communication between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main elements of SPI and their interactions are shown in the following block diagram at *Figure 571*.

Figure 571. SPI2S block diagram



The simplified scheme of *Figure 571* shows three fully independent clock domains:

- The **spi_pclk** clock domain,
- The **spi_ker_ck** kernel clock domain,
- The serial interface clock domain,

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed in order to avoid any data underrun or overrun events only.

The **spi_pclk** clock signal feeds the peripheral bus interface. It has to be active when it accesses to the SPI registers are required.

The SPI master needs the **spi_ker_ck** kernel clock coming from RCC active during communication to feed the serial interface SCK clock via the clock generator divider.

The SPI working in slave mode handles data flow using the serial interface clock derived from the external SCK signal provided by external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive.

As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be setup correctly (e.g. when underrun or overrun is evaluated see *Section 52.5.2: SPI error flags* for details). This cannot be done when the bus becomes into idle. At specific case the slave even requires the clock generator working (see *Section 52.5.1: TI mode*).

52.4.2 SPI signals

Four I/O pins are dedicated to SPI communication with external devices.

- **MISO:** Master In / Slave Out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** Master Out / Slave In data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** Serial Clock output pin for SPI masters and input pin for SPI slaves.
- **SS:** Slave select pin. Depending on the SPI and SS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame or
  - Detect a conflict between multiple masters

  See *Section 52.4.7: Slave select (SS) pin management* for details.

The SPI bus allows the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals can be added depending on the data exchange between SPI nodes and their slave select signal management. the functionality between MOSI and MISO pins can be inverted in any SPI mode (see the IOSWP bit at SPI_CFG2 register).

All these pins are shared with I2S if this mode is implemented at the instance. See *Section 52.9.2: Pin sharing with SPI function*.

52.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use 2 or 3 wires (with software SS management) or 3/4 wires (with hardware SS management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slave(s) for communication by SS line when it is managed by HW. The data between the master and the slave flow on the MOSI and/or MISO lines. The direction of data flow is highlighted by black arrows at the following topology figures.

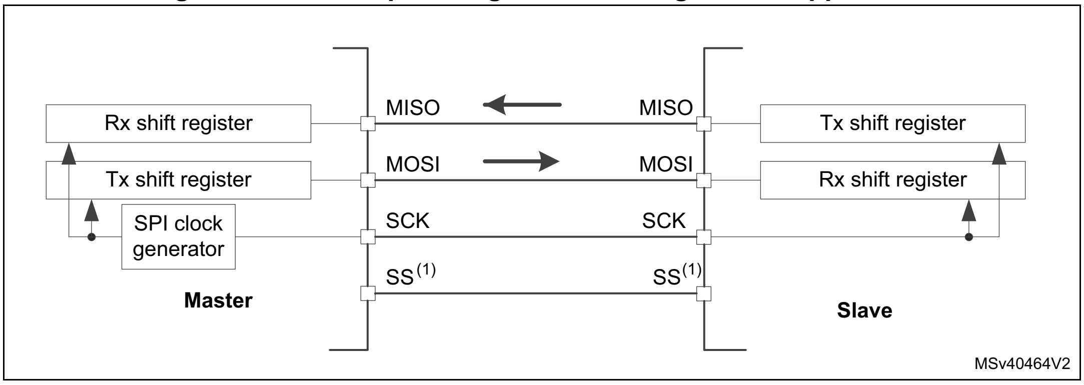
52.4.4 Communications between one master and one slave

The communication flow may use one of 3 possible modes: full-duplex (3 wires), half-duplex (2 wires) or simplex (2 wires). The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS signal can be helpful at this configuration to synchronize the data flow and it is used by default at some specific SPI modes (e.g. TI mode).

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0]=00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 572. Full-duplex single master/ single slave application

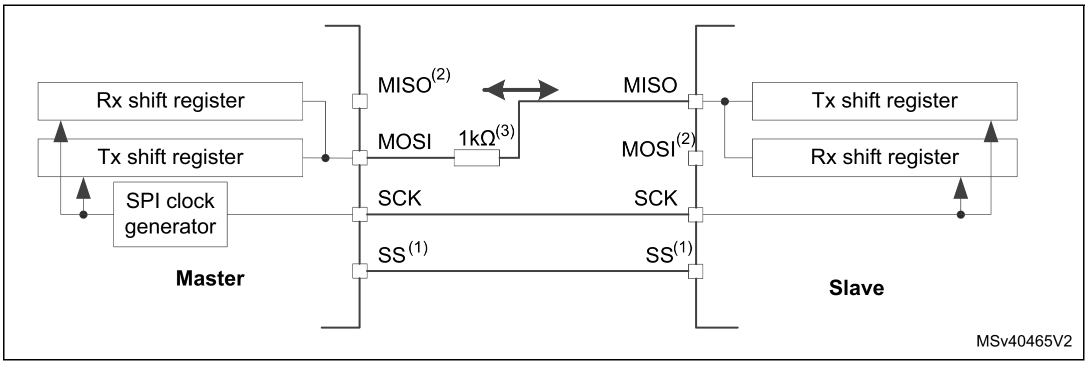


1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see *Section 52.4.7: Slave select (SS) pin management* for details).

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0]=11 in the SPI_CFG2 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI has to be disabled when changing direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 573. Half-duplex single master/ single slave application



1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see *Section 52.4.7: Slave select (SS) pin management* for details).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when communication direction is changed not synchronously between two nodes working at bidirectional mode and new transmitter accesses the common data line while former transmitter still keeps an opposite value on the line (the value depends on SPI configuration and communicated data). Both nodes can fight with opposite outputs levels on the line temporary till next node change its direction setting correspondingly, too. It is suggested to insert serial resistance between MISO and MOSI pins at this mode to protect the outputs and limit the current blowing between them at this situation,

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pins pair is not used for communication and can be used as standard GPIOs.

- Transmit-only mode: COMM[1:0]=01

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is on-going.

  The slave in transmit only mode sends data as long as it receives a clock on the SCK pin and the SS pin (or SW managed internal signal) is active (see *Section 52.4.7: Slave select (SS) pin management*).

- **Receive-only mode: COMM[1:0]=10**

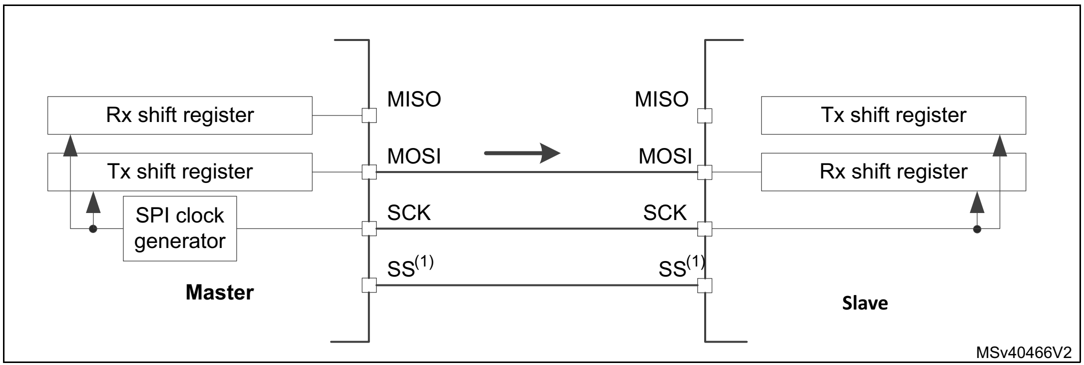
  In master mode, the MOSI output is disabled and may be used as GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by SW explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 52.4.7: Slave select (SS) pin management*).

Note: At whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register. (This bit may only be modified when the SPI is disabled).
Any simplex communication can be replaced by a variant of the half duplex communication

with a constant setting of the transaction direction (bidirectional mode is enabled, while the HDDIR bit is never changed).

Figure 574. Simplex single master/single slave application (master in transmit-only/ slave in receive-only mode)

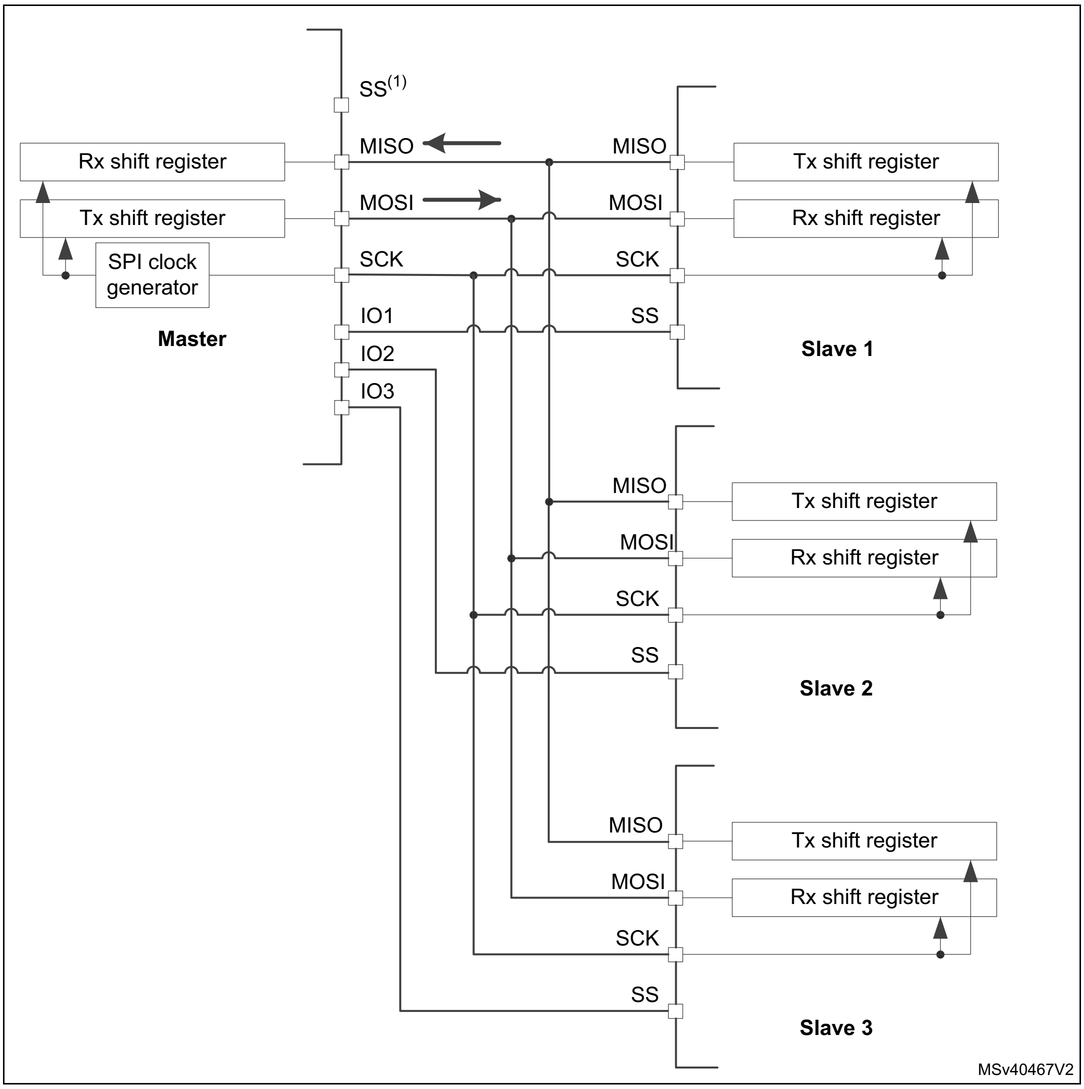


1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see *Section 52.4.7: Slave select (SS) pin management* for details).
2. In this configuration, both the MISO pins can be used as GPIOs.

52.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 575.*). The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS input (only one slave can control data on common MISO line at time). When this is done, a communication between the master and the selected slave is established. Except the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

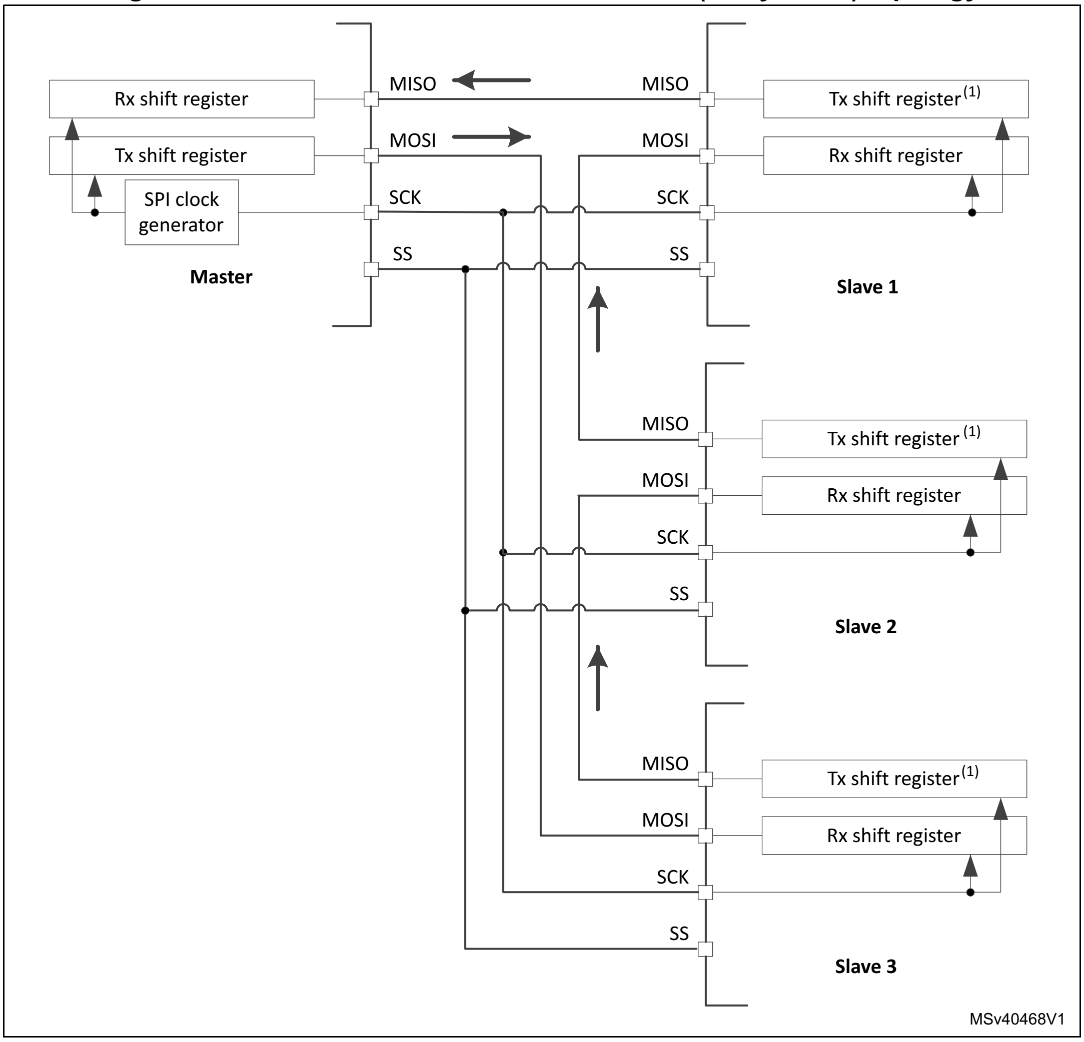
Figure 575. Master and three independent slaves at star topology



1. Master single SS pin hardware output functionality cannot support this topology (to be replaced by set of GPIOs under SW control) and user should avoid SPI AF setting at the pin (see *Section 52.4.7: Slave select (SS) pin management* for details).
2. If the application cannot ensure that no more than a single SS active signal is provided by the master at time, it is better to configure the MISO pins into open drain configuration with an external pull-up at MISO line to prevent any conflict between interconnected outputs of the slaves on the line. Else the push-pull configuration can be applied without an extra resistor for the slaves. (see *Section 13.3.7: I/O alternate function input/output on page 1063*).

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see *Figure 576*). All the slaves behave like simple shift registers applied at serial chain under common slave select and clock control. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transacted by the master is equal to the number of slaves. Then when a first data frame is transacted in the chain, the master just sends information dedicated for the last slave node in the chain via the first slave node input while the first information received by the master comes from the last node output at this time. Correspondingly, the lastly transacted data finishing the session is dedicated for the first slave node while its firstly outgoing data just reaches the master input after its circling around the chain passing through all the other slaves during the session. The data format configuration and clock setting has to be the same for all the nodes in the chain at this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun has to be applied at the TxFIFO slaves when information is transacted between the receiver and the transmitter by hardware. In this case, the transmission underrun feature is configured at a mode repeating lastly received data frame (UDRCFG[1:0]=01). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case the underrun happens in fact after this first data frame is transacted (underrun detection has to be set at end of data transaction at slaves UDRDET[1:0]=01). To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user has to disable and enable the SPI between sessions and fill the TxFIFO by a new single data pattern.

Figure 576. Master and three slaves at circular (daisy chain) topology



1. Underrun feature is used at slaves at this configuration when slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.

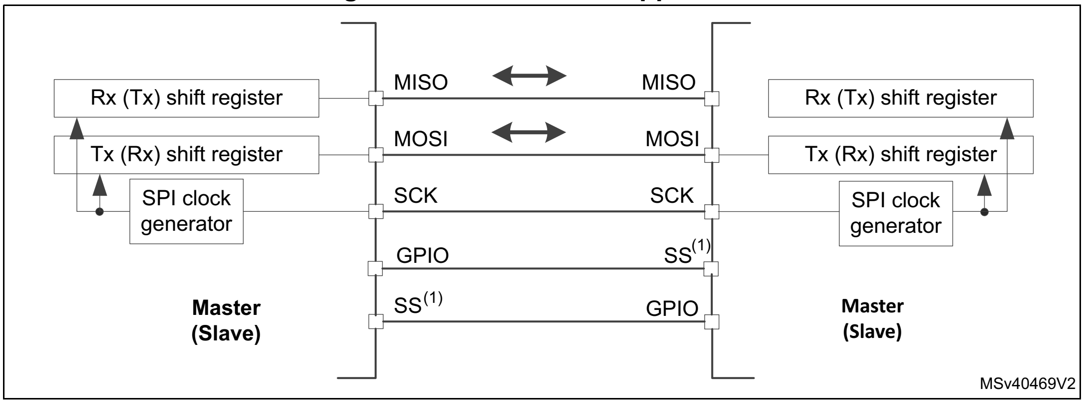
52.4.6 Multimaster communication

Unless the SPI bus is not designed for a multimaster capability primarily, the user can use build in feature which detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the SS pin is used configured at hardware input mode. The connection of more than two SPI nodes working at this mode is impossible as only one node can apply its output on a common data line at time.

When nodes are non active, both stay at slave mode by default. Once one node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for next session start.

If potentially both nodes raised their mastering request at the same time a bus conflict event appears (see mode fault MODF event). Then the user can apply some simple arbitration process (e.g. to postpone next attempt by predefined different time-outs applied at both nodes).

Figure 577. Multimaster application



1. The SS pin is configured at hardware input mode at both nodes. Its active level enable the MISO line output control as passive node is configured as a slave.

52.4.7 Slave select (SS) pin management

In slave mode, the SS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the SS can be used either as an output or an input. As an input it can prevent a multi master bus collision, and as an output it can drive a slave select signal of a single slave. The SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active one by the SSIOP bit setting. The SS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software SS management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the register SPI_CR1. The external SS pin is free for other application uses (as GPIO or other alternate function).
- **Hardware SS management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in register SPI_CFG2).
  - **SS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The SS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply proper TSIZE>0 setting to control the SS output correctly. Even if SPI AF is not applied at the SS pin (it can be used as a standard GPIO then), SSOE=1 setting should be kept anyway to assure default SS input level and prevent any mode fault evaluation at input of the master SS internal logic applicable at a multimaster topology exclusively.

    a) When SSOM = 0 and SP = 000, the SS signal is driven to the active level as soon as the master transfer starts (CSTART=1) and it is kept active until its EOT flag is set or the transmission is suspended.

    b) When SP = 001, a pulse is generated as defined by the TI mode.

    c) When SSOM=1, SP=000 and MIDI>1 the SS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (1 to 14).
  - **SS output disable (SSM=0, SSOE = 0)**:

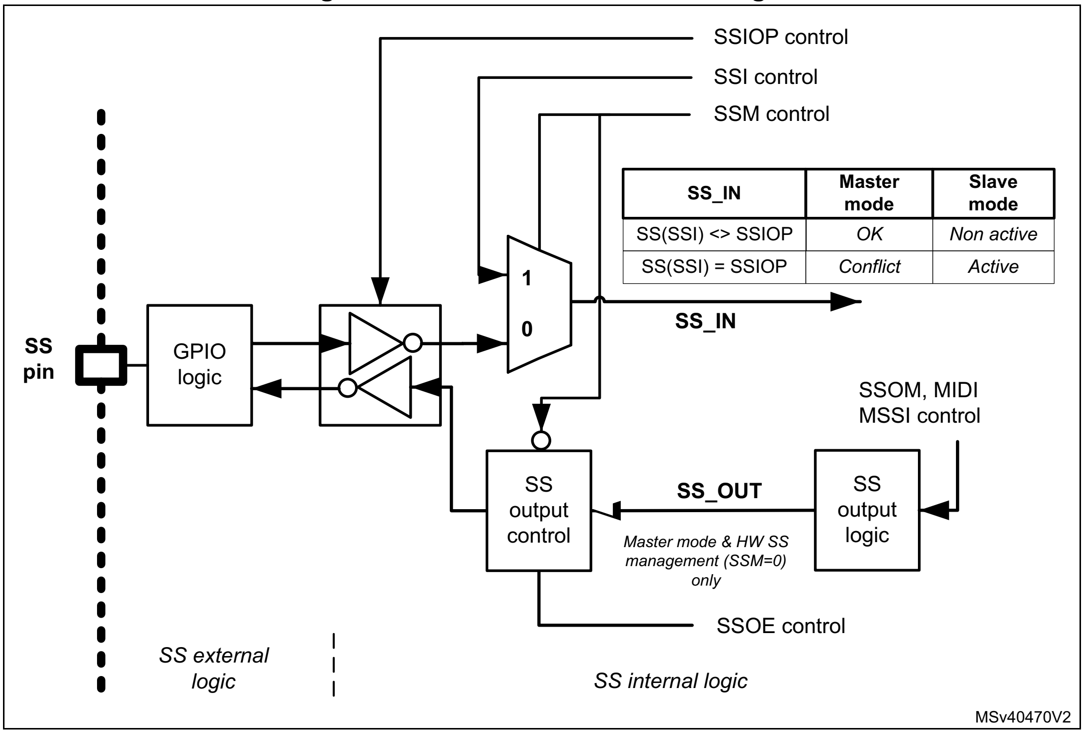
    a) if the microcontroller is acting as the master on the bus, this configuration allows multi master capability. If the SS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER=0).

    b) In slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS line is at its active level.

Note: The purpose of automatic switching into Slave mode at mode fault condition is to avoid the possible conflicts on data and clock line. As the SPE is automatically reset at this condition, both Rx and Tx FIFOs are flushed and current data is lost.

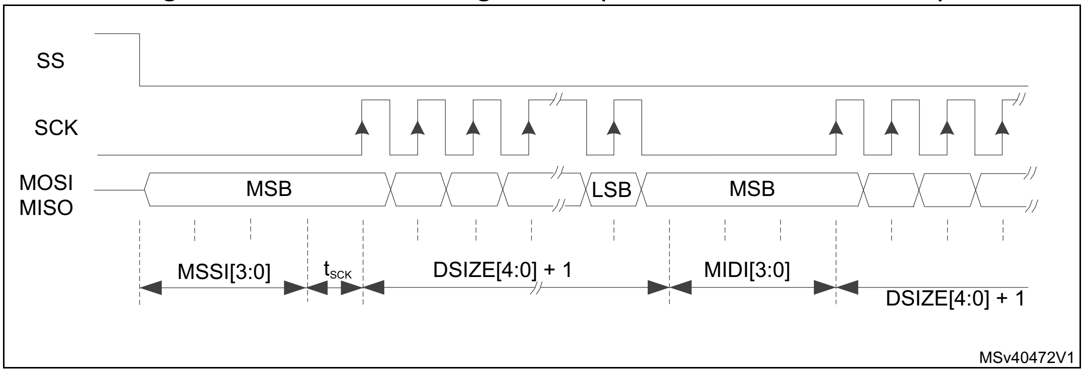
Note: When the SPI slave is enabled at the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level till the slave detects a start of the SS signal (its transition from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection at the SS software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

Figure 578. Scheme of SS control logic



When a hardware output SS control is applied (SSM=0, SSOE=1), by configuration of MIDI[3:0] and MSSI[3:0] bitfields the user can control timing of the SS signal between data frames and insert an extra delay at begin of every transaction (to separate the SS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 579: Data flow timing control (SSOE=1, SSOM=0, SSM=0)*

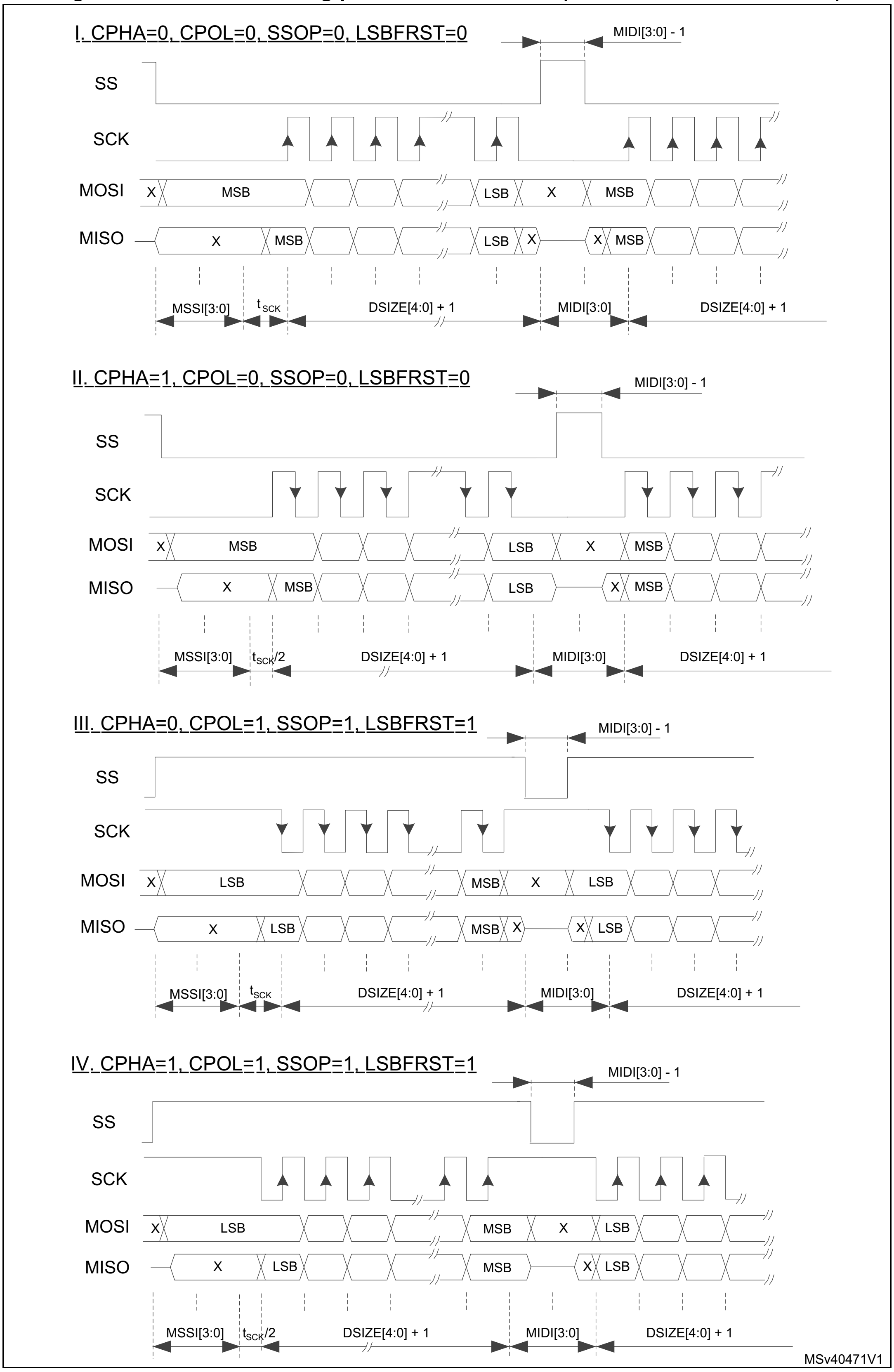
Figure 579. Data flow timing control (SSOE=1, SSOM=0, SSM=0)



1. MSSI[3:0]=0011, MIDI[3:0]=0011 (SCK flow is continuous when MIDI[3;0]=0).
2. CPHA=0, CPOL=0, SSOP=0, LSBFRST=0.

Additionally, bit SSOM=1 setting invokes specific mode which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] has to be set greater then one SPI period). Some configuration examples are shown at *Figure 580: SS interleaving pulses between data (SSOE=1, SSOM=1,SSM=0)*.

Figure 580. SS interleaving pulses between data (SSOE=1, SSOM=1,SSM=0)



1. MSSI[3:0]=0010, MIDI[3:0]=0010.
2. SS interleaves between data when MIDI[3:0]>1.

52.4.8 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data are being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

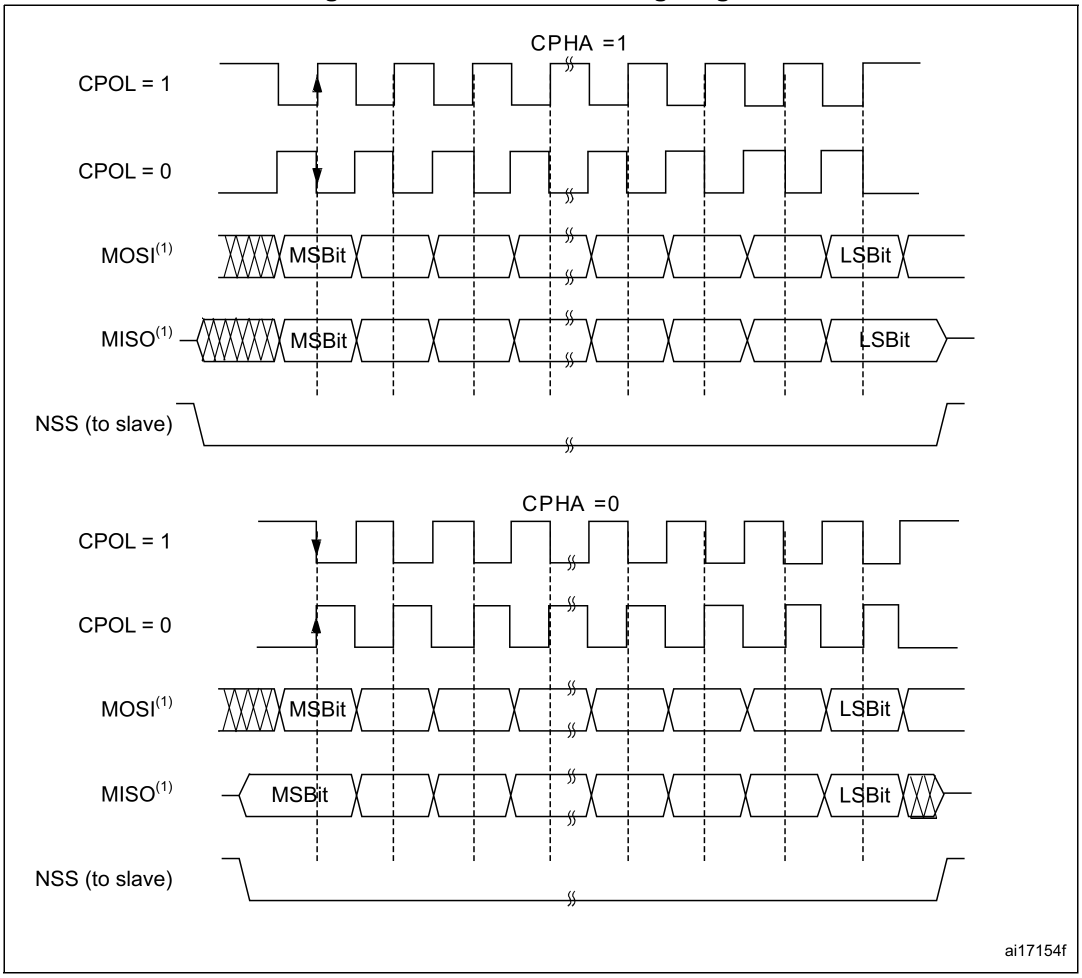
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines at *Figure 581: Data clock timing diagram*).

Figure 581, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL=1 or pulling it down if CPOL=0).

Figure 581. Data clock timing diagram



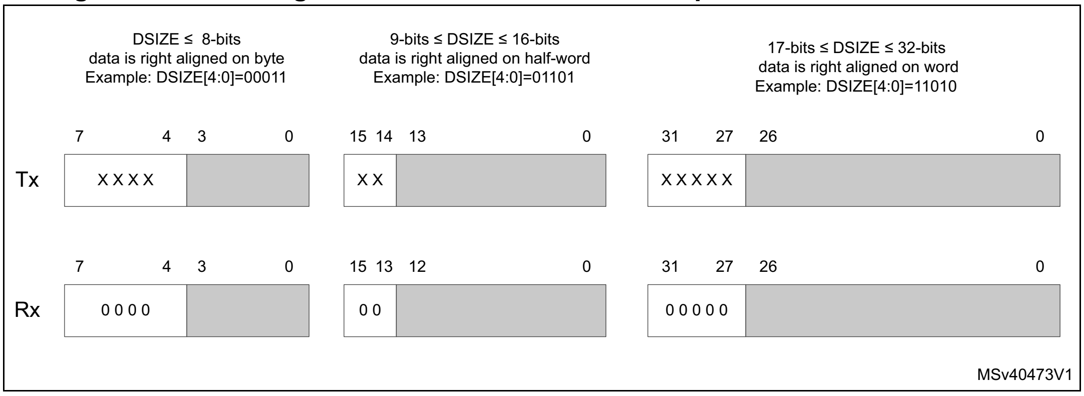
1. The order of data bits depends on LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit in SPI_CFG2 register. The data frame size is chosen by using the DSIZE[4:0] bits. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. When the SPI_TXDR/SPI_RXDR registers are accessed, data frames are always right-aligned into either a byte (if the data fit into a byte), a half-word or a word (see *Figure 582*).

If the access is a multiple of the configured data size, data packing is applied automatically. During communication, only bits within the data frame are clocked and transferred.

Figure 582. Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit



Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 4-bit data frame size.

52.4.9 Configuration of SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication has to be initialized, perform these steps prior SPI is enabled:

1. Write the proper GPIO registers: Configure GPIO alternate functions at MOSI, MISO, SCK and SS pins if applied.
2. Write to the SPI_CFG1 and SPI_CFG2 registers to set up proper values of all not reserved bits and bitfields included there with next exceptions:
   a) SSOM, SSOE, MBR[2:0], MIDI[3:0] and MSSI[3:0] are required and taken into account at master mode exclusively.
   b) UDRDET[1:0] and UDRCFG[1:0] are required and taken into account at slave mode only. The MBR[2:0] setting is taken into account only when slave is configured at TI mode.
   c) CRCSIZE[4:0] is required if CRCEN is set,
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, MSSI, MIDI and SSM are not required at TI mode.
   e) Once the AFCNTR bit is set at SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless the peripheral enable so any later configurations changes of the SPI_CFG1 and SPI_CFG2 registers can affect level of signals at these pins.
   f) The I2SMOD bit at SPI_I2SCFGR register has to be kept cleared to prevent any unexpected influence of occasional I2S configuration.
3. Write to the SPI_CR2 register to select length of the transfer, if it is not known TSIZE has to be programmed to zero.
4. Write to SPI_CRCPOLY and into TCRCINI, RCRCINI and CRC33_17 bits at SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see chapter Communication using DMA).
6. Configure SSI, HDDIR and MASRX at SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CFG1 register if the configuration protection is required (for safety).

52.4.10 Procedure for enabling SPI

It is recommended to configure and enable the SPI slave before the master sends the clock but there is no impact if the configuration and enabling procedure is done while a traffic is ongoing on the bus suppose SS signal is managed by hardware at slave or kept inactive by slave software when software management of the SS signal is applied (see *Section 52.4.7: Slave select (SS) pin management*). The data register of the slave transmitter must contain data to be sent before the master starts its clocking.The SCK signal must be settled to idle state level corresponding to the selected polarity before the SPI slave is selected by SS else following transaction may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode all the traffics are ignored even in case of the SS is found at active level till the slave detects a start of the SS signal (its transition from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when external SS pin is fixed. There is no such protection at the SS software management. In this case the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master at full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 52.4.14: Communication using DMA (direct memory addressing)*.

52.4.11 SPI data transmission and reception procedures

The setting of data communication format follows the basic principle that sure number of data with a flexible size must be transferred within a session (transaction) while, optionally, the data handling can be cumulated effectively into a single access of the SPI data registers (data packing) or even grouped into a sequence of such services if data is collected at consistent bigger data packets. The data handling services are based upon FIFO packet occupancy events. That is why the complete data packet must be serviced exclusively upon a dedicated packet flag.

To understand better the next detailed content of this section, the user should capture the configuration impact and meaning of the following items at first:

Data size (DSIZE)- defines data frame (sets the number of bits at single data frame).

FIFO threshold (FTHLV) - defines data packet, sets the number of data frames at single data packet and so the occurrence of the packet occupancy events to handle SPI data registers either by software or by DMA.

Data access – a way how to handle the SPI data register content when the transfer data between the application and the SPI FIFOs upon a packet event. It depends on the packet size configuration. Optionally, multiply data can be handled effectively by a single access of the register (by data packing) or by sequence of such accesses (when servicing a bigger data packet).

FIFO size – capacity or space to absorb available data. It depends on the data size and the internal hardware efficiency how the data is compressed and organized within this space. The FTHLV setting must respect the FIFO capacity to store two data packets at least.

Transaction size (TSIZE) – defines total number of data frames involved at a transaction session overall possibly covered by several data packet services. There is no need to align this number with the packet size (handling of a last not aligned data packet is supported if TSIZE is programmed properly).

Data handling via RxFIFO and TxFIFO

All SPI data transitions pass through the embedded FIFOs organized by bytes (N x 8-bit). The size of the FIFOs (N) is product and the peripheral instance dependent. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short or the interrupt/DMA latency is too long. Each direction has its own FIFO called TxFIFO and RxFIFO, respectively.

The handling of the FIFOs content is based on servicing data packet events exclusively raised by dedicated FIFO packet occupancy flags (TXP, RXP or DXP). The flags occurrence depends on the data exchange mode (duplex, simplex), the data frame size (number of bits in the frame) and how data are organized at data packets. The frequency of the packet events can be decreased significantly when data are organized into packets via defining the FIFOs threshold. Several data frames grouped at packet can be then handled effectively based on a single FIFO occupancy packet event either by a single SPI data register access or their sequence what consumes less system performance. The user can control the access type by casting the data register address to force a concrete CPU instruction applied for the register read or write. The access then can be 8-bit, 16-bit or 32-bit but single data frame must be always accessed at least. It is crucial to keep the setting of the packet size (FTHLVL) and the data size (DSIZE) always balanced with the applied data registers access (no matter if a single access or their sequence is applied) just to apply and complete service of a single data packet upon its event. This principle, occurrence and clearing capabilities of the FIFO occupancy flags are common no matter if DMA, interrupt, or polling is applied.

A read access to the SPI_RXDR register returns the oldest value stored in the RxFIFO that has not been read yet. A write access to the SPI_TXDR stores the written data in the TxFIFO at the end of a send queue.

A read access to the SPI_RXDR register must be managed by the RXP event. This flag is set by hardware when at least one complete data packet (defined as receiver threshold by FTHLV[3:0] bits at the SPI_CFG1 register) is available at the reception FIFO while reception is active. The RXP is cleared as soon as less data than a complete single packet is available in the RxFIFO, when reading SPI_RXDR by software or by DMA.

The RXP triggers an interrupt if the RXPIE bit is set.

Upon setting of the RXP flag, the application performs the due number of SPI data register reads to download the content of one data packet. Once a complete data packet is downloaded, the application software or DMA checks the RXP value to see if other packets are pending into the receive FIFO and, if so, downloads them packet by packet until the RXP reads 0. RxFIFO can store up to N data frames (for frame size =< 8-bit), N/2 data frames (for 8-bit < frame =< 16-bit), N/3 data frames (for 16-bit < frame =< 24-bit) or N/4 data frames (if data frame >24-bit) where N is the size of the FIFO in bytes.

At the end of a reception, it may happen that some data may still be available in the RxFIFO, without reaching the FTHLV level, thus the RXP is not set. In this case, the number of remaining RX data frames in the FIFO is indicated by RXWNE and RXPLVL fields in the SPI_SR register. It happens when number of the last data received in a transfer cannot fully accomplish the configured packet size in the case transfer size and packet size are not aligned. Nevertheless the application software can still perform the standard number of reads from the RxFIFO used for the previous complete data packets without drawbacks:

only the consistent data (completed data frames) are popped from the RxFIFO while redundant reads (or any uncompleted data) are reading 0. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the reception of the last data in a transfer and from calculating the due number of reads to be popped from RxFIFO.

In a similar way, write access of a data frame to be transmitted is managed by the TXP event. This flag is set by hardware when there is enough space for the application to push at least one complete data packet (defined at FTHLV[3:0] bits at SPI_CFG1 register) into the transmission FIFO while transmission is active. The TXP is cleared as soon as the TxFIFO is filled by software a/o by DMA and space currently available for any next complete data packet is lost. This can lead to oscillations of the TXP signal when data are released out from the TxFIFO while a new packet is stored frame by frame. Any write to the TxFIFO is ignored when there is no sufficient room to store at least a single data frame (TXP event is not respected), when TXTF is set or when the SPI is disabled.

The TXP triggers an interrupt if the TXPIE bit is set or a/o a DMA request if TXDMAEN is set. The TXPIE mask is cleared by hardware when the TXTF flag is set.

Upon setting of the TXP flag application software performs the due number of SPI data register writes to upload the content of one entire data packet. Once new complete data packet is uploaded, the application software or DMA checks the TXP value to see if other packets can be pushed into the TxFIFO and, if so, uploads them packet by packet until TXP reads 0 at the end of any packet load.

The number of last data in a transfer can be shorter than the configured packet size in the case when the transfer size and the packet size are not aligned. Nevertheless the application can still perform the standard number of data register writes used for the previous packets without drawbacks: only the consistent data are pushed into the TxFIFO while redundant writes are discarded. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the transmission of the last data in a transfer and from calculating the due number of writes to push the last data into TxFIFO. Just for the last data case, the TXP event is asserted by SPI once there is enough space into TxFIFO to store remaining data to complete current transfer.

Both TXP and RXP events can be polled or handled by interrupts. The DXP bit can be monitored as a common TXP and RXP event at full duplex mode.

Upon setting of the DXP flag the application software performs the due number of writes to the SPI data register to upload the content of one entire data packet for transmission, followed by the same number of reads from the SPI data register to download the content of one data packet. Once one data packet is uploaded and one is downloaded, the application software or DMA checks the DXP value to see if other packets can be pushed and popped in sequence and, if so, uploads/downloads them packet by packet until DXP reads 0.

The DXP triggers an interrupt if the DXPIE bit is set. The DXPIE mask is cleared by hardware when the TXTF flag is set.

The DXP is useful in Full-Duplex communication in order to optimize performance in data uploading/downloading, and reducing the number of interrupts or DMA sequences required to support an SPI transfer thus minimizing the request for CPU bandwidth and system power especially when SPI is operated in Stop mode.

When relay on the DXP interrupt exclusively, the user must consider the drawback of such a simplification when TXP and RXP events are serviced by common procedures because the TXP services are delayed by purpose in this case. This is due to fact that the TXP events precedes the reception RXP ones normally to allow the TXP servicing prior transaction of the last frame fully emptying the TxFIFO else master cannot provide a continuous SCK clock flow and the slave can even face an underrun condition. The possible solution is to prefill the TxFIFO by few data packets ahead prior the session starts and to handle all the data received after the TXTF event by EOT exclusively at the end of the transaction (as TXTF suppresses the DXP interrupts at the end of the transaction). In case of CRC computation is enabled, the user must calculate with additional space to accommodate the CRC frame at RxFIFO when relying on EOT exclusively at the end of transaction.

Another way to manage the data exchange is to use DMA (see *Communication using DMA (direct memory addressing)*).

If the next data is received when the RxFIFO is full, an overrun event occurs (see description of OVR flag at *Section 52.5.2: SPI error flags*). An overrun event can be polled or handled by an interrupt.

This may happen in slave mode or master mode (full duplex or receive only with MASRX = 0). In master receive only mode, with MASRX = 1, the generated clock stops automatically when the RxFIFO is full, therefore overrun is prevented.

Both RxFIFO and TxFIFO content is kept flushed when SPI is disabled (SPE=0).

Transaction handling

A few data frames can be passed at single transaction to complete a message. The user can handle number of data within a message thanks to values stored into TSIZE and TSER fields. In principle, the transaction of a message starts when the SPI is enabled by setting CSTART bit and finishes when the total number of required data is transacted. The end of transaction controls the CRC and the hardware SS management when applied. To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed. If TSIZE is kept at zero while CSTART is set, an endless transaction is initialized (no control of transfer size is applied). During an endless transaction, the number of transacted data aligned with FIFOs threshold is supported exclusively. If the number of data (or its grouping into packets) is unpredictable, the user must keep the FIFO threshold setting (packet size) at single data (FTHLV=0) to assure that each data frame raises its own packet event to be serviced by the application or DMA.

The transaction can be suspended at any time thanks to CSUSP which clears the CSTART bit. SPI must be always disabled after such software suspension and re-enabled before the next transaction starts.

In master mode, the user can extend the number of data within the current session. When the number of data programmed into TSIZE is transacted and if TSER contains a non-zero value, the content of TSER is copied into TSIZE, and TSER value is cleared automatically. The transaction is then extended by a number of data corresponding to the value reloaded into TSIZE. The EOT event is not raised in this case as the transaction continues. After the reload operation, the TSERF flag is set and an interrupt is raised if TSERFIE is set. The user can write the next non-zero value into TSER after the TSER is cleared by hardware but still before the next reload occurs, so an unlimited number of data can be transacted while repeating this process.

When any data extension is applied, it always starts by aligned data packet. That is why it is suggested to keep number of data to be extended always aligned with packet size else the last data packet just before the extension is applied has to be handled as an incomplete one (see data packing chapter). If overall number of data is not aligned, the user must implement the rest not aligned number of data into TSER just at the last extension cycle and then handle the last incomplete packet of data standardly within EOT event handler.

For example, if the user wants to transfer 23 bytes while applies data number extension at configuration of 8-bit data size, data packet set to 4 data and 32-bit access to FIFO is used then whatever next sequence is correct

– TSIZE=16 TSER=7;
– TSIZE=12 TSER=8; last extensionTSER=3;

As the last not aligned MSB byte is ignored just within the last (6th) access of the FIFO.

When a not aligned sequence is applied for data to be extended like at the following cases

– TSIZE=15 TSER=8 or
– TSIZE=8 TSER=7; last extension TSER=8;

The MSB byte is ignored within the 4th access of the FIFO while the other accesses handle always 4 data at the FIFO.

When the transmission is enabled, a sequence begins and continues while any data is present in the TxFIFO of the master. The clock signal is provided permanently by the master until TxFIFO becomes empty, then it stops, waiting for additional data.

In receive-only modes, half duplex (COMM[1:0]=11, HDDIR=0) or simplex (COMM[1:0]=10) the master starts the sequence when SPI is enabled and transaction is released by setting the CSTART bit. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled/suspended by the master. The master receives data frames permanently up to this moment. The reception can be suspended either by SW control, writing 1 to the CSUSP bit in the SPI_CR1 register, or automatically when MASRX=1 and RxFIFO becomes full. The reception is automatically stopped also when the number of frames programmed in TSIZE and TSER fields of the SPI_CR2 register has been completed.

In order to disable the master receive only mode, the SPI must be suspended at first. When the SPI is suspended, the current frame is completed, before changing the configuration.

Caution: If SPE is written to 0 at master, while reception is ongoing without any suspending, the clock is stopped without completing the current frame, and the RxFIFO is flushed.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it has to respect slave capability to handle data flow and its content at anytime. When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays by MIDI[3:0] bits setting or provide an initial delay by setting MSSI[1:0] which postpones any transaction start to give slave sufficient room for preparing data. Be aware data from the slave are always transacted and processed by the master even if the slave could not prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are short, FIFO is accessed by bytes and the SPI bus rate is high.

In order to add some SW control on the SPI communication flow from a slave transmitter node, a specific value written in the SPI_UDRDR (SPI Underrun Data Register) may be used. On slave side, when TxFIFO becomes empty, this value is sent out automatically as next data and may be interpreted by SW on the master receiver side (either simply dropped or interpreted as a XOFF like command, in order to suspend the master receiver by SW).

At multislave star topology, a single slave can be only enabled for the output data at a time. The slave just selected for the communication with the master needs to detect a change of its SS input into active level before the communication with the master starts. In a single slave system it is not necessary to control the slave with SS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. The SS can be managed by both software and hardware (*Section 52.4.7: Slave select (SS) pin management*).

52.4.12 Procedure for disabling the SPI

When SPI is disabled, it is mandatory to follow the disable procedures described in this paragraph.

At the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped. Otherwise, ongoing transactions may be corrupted in this case.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into STOP mode. Refer to the RCC section for further information.

The master in full duplex or transmit only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. TXC flag can be polled (or interrupt enabled with EOTIE=1) in order to wait for the last data frame to be sent.

When the master is in any receive only mode, in order to stop the peripheral, the SPI communication must be first suspended, by setting CSUSP to 1.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user has to ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When the master handles SS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide proper end of SS pulse for slave, or
- When transaction streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

When TSIZE>0, EOT and TXC signals are equal so the polling of EOT is reliable at whatever SPI communication mode to check end of the bus activity. When TSIZE=0, the user has to check TXC, SUSP or FIFO occupancy flags in according with the applied SPI mode and way of the data flow termination.

The correct disable procedure in master mode, except when receive only mode is used, is:

1. Wait until TXC=1 and/or EOT=1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when CRC frame is completed in this case. When a transmission is suspended the software has to wait till CSTART bit is cleared.
2. Read all RxFIFO data (until RXWNE=0 and RXPLVL=00)
3. Disable the SPI (SPE=0).

The correct disable procedure for master receive only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP=1)
2. Wait until SUSP=1 (the last data frame is processed) if receive flow is suspended.
3. Read all RxFIFO data (until RXWNE=0 and RXPLVL=00)
4. Disable the SPI (SPE=0).

In slave mode, any on going data are lost when disabling the SPI.

52.4.13 Data packing

From user point of view there are two ways of data packing which can overlay each other:

- Type of access when data are written to TxFIFO or read from RxFIFO
  Multiple data can be pushed or fetched effectively by single access if data size is multiplied less than access performed upon SPI_TXDR or SPI_RXDR registers.
- Number of data to be handled during the single software service
  It is convenient to group data into packets and cumulate the FIFO services overall the data packet content exclusively instead of handling data frame by frame separately. The user can define packets by FIFO threshold settings. Then all the FIFO occupancy events are related to that threshold level while required services are signalized by proper flags with interrupt and/or wake up capabilities.

When the data frame size fits into one byte (less than or equal to 8 bits), the data packing is used automatically when any read or write 16-bit or 32-bit access is performed on the SPI_RXDR/SPI_TXDR register. The multiple data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other data stored in the MSB. *Figure 583* provides an example of data packing mode sequence handling. While DSIZE[3:0] is configured to 4-bit there, two or four data frames are written in the TxFIFO after the single 16-bit or 32-bit access the SPI_TXDR register of the transmitter.

When the data frame size is between 9-bit and 16-bit, data packing is used automatically when a 32-bit access is done. the least significant half-word is used first. (regardless of the LSBFRST value)

This sequence can generate two or four RXP events in the receiver if the RxFIFO threshold is set to 1 frame (and data is read on a frame basis, unpacked), or it can generate a single RXP event if the FTHLV[3:0] field in the SPI_CFG1 register is programmed to a multiple of the frames to be read in a packed mode (16-bit or 32-bit read access).

The data are aligned in accordance with *Figure 582: Data alignment when data size is not equal to 8-bit, 16-bit or 32-bit*. The valid bits are performed on the bus exclusively. Unused bits are not cared at transmitter while padded by zeros at receiver.

When short data frames (<8-bit or < 16-bit) are used together with a larger data access mode (16-bit or 32-bit), the FTHLV value must be programmed as a multiple of the number of frames/data access (i.e. multiple of 4 if 32-bit access is used to up to 8-bit frames or multiple of 2 if 16-bit access is used to up to 8-bit frames or 32-bit access to up to 16-bit frames.).

The RxFIFO threshold setting must always be higher than the following read access size, as spurious extra data would be read otherwise.

The FIFO data access less than the configured data size is forbidden. One complete data frame has to be always accessed at minimum.

A specific problem appears if an incomplete data packet is available at FIFO: less than 4x8-bit frames or one single 16-bit frame is available.

There are two ways of dealing with this problem:

A. without using TSIZE field

On transmitter side, writing the last data frame of any odd sequence with an 8-bit/16-bit access to SPI_TXDR is enough.

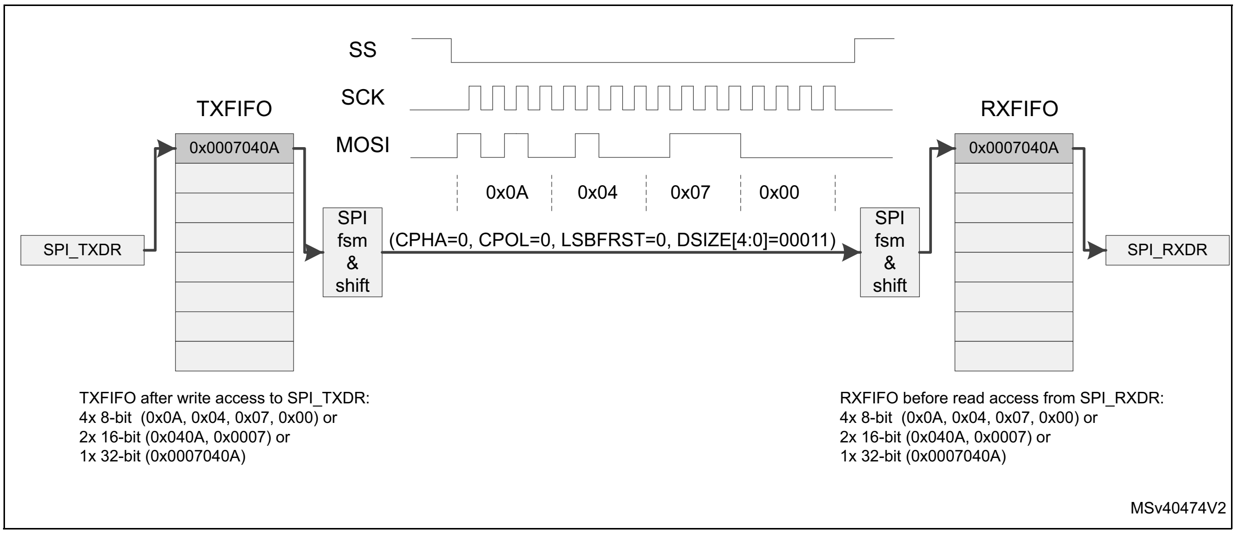
On receiver side, the remaining data may be read by any access. Any extra data read are padded with zeros. Polling the RXWNE and RXPLVL may be used to detect when the RX data are available in the RxFIFO. (a time out may be used at system level in order to detect the polling)

B. using the TSIZE field

On transmitter side, the transaction is stopped by the master when it faces EOT event.

In reception, the RXP flag is not set when EOT is set. In the case when the number of data to be received (TSIZE) is not a multiple of packet size, the number of remaining data is indicated by the RXWNE and RXPLVL fields in the SPI_SR register. The remaining data can be read by any access. Any extra read is padded by zeros.

Figure 583. Packing data in FIFO for transmission and reception



1. DSIZE[3:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST, if it is set, the order is reversed at all the data frames.

52.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits in the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set to 1. The DMA then performs series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set to 1. The DMA then performs series of reads from the SPI_RXDR register. When EOT is set at the end of transaction and last data packet is incomplete then DMA request is activated automatically in according with RXWNE and RXPLVL[1:0] setting to read rest of data.

If the SPI is programmed in receive only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel and disabled.

If the SPI is programmed in transmit only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set at SPI_SR register), the EOT (or TXC at case TISEZE=0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT=1 and/or TXC=1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:

1. Disable DMA request for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

If the DMA channel PSIZE value is equal to 16-bit or 32-bit and SPI data size is less than or equal to 8-bit, then packing mode is enabled. Similarly, If the DMA channel PSIZE value is equal to 32-bit and SPI data size is less than or equal to 16-bit, then packing mode is enabled.The DMA then automatically manages the write operations to the SPI_TXDR register.

Regardless data packing mode is used and the number of data to transfer is not a multiple of the DMA data size (16-bit or 32-bit) while the frame size is smaller, DMA completes the transfer automatically in according with the TSIZE field setting.

Alternatively, last data frames may be written by software, in the single/unpacked mode.

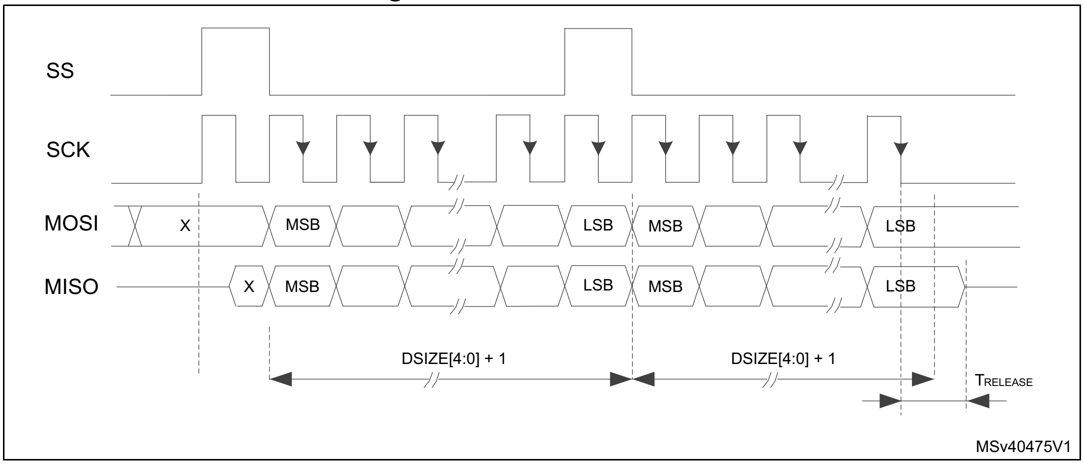
To configure any DMA data access less than the configured data size is forbidden. One complete data frame has to be always accessed at minimum.

52.5 SPI specific modes and control

52.5.1 TI mode

By specific setting of the SP[2:0] bit field at the SPI_CFG2 register the SPI can be configured to be compliant with TI protocol. The SCK and SS signals polarity, phase and flow as well as the bits order are fixed so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP and SSM is not required when the SPI is at TI mode configuration. The SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown at the *Figure 584: TI mode transfer*.

Figure 584. TI mode transfer



In slave mode, the clock generator is used to define time when the slave output at MISO pin becomes to HiZ when the current transaction finishes. The master baud rate setting (MBR[2:0] at SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become HiZ (TRELEASE) depends on internal re-synchronization, too, which takes next additional 2-4 periods of the clock signal feeding the generator. It is given by formula:

$$\frac{Tbaud}{2} + 2 \times Tspi\_ker\_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi\_ker\_ck$$

If the slave detects misplaced SS pulse during data transaction the TIFRE flag is set.

52.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the corresponding Interrupt Enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. OVR flag triggers an interrupt if OVRIE bit is set. Clearing the OVR bit is done by a writing 1 to the OVRC bit in the SPI_IFCR. To prevent any next overrun event the clearing must be done after RxFIFO is emptied by software reads. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on the RXP flag indication. At master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

At a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO at the moment specified by UDRDET bits. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Once the underrun is captured next provided data for transmission depends on the UDRCFG bits. The slave can provide out either data stored lastly to its TxFIFO or the data received previously from the master or a constant pattern stored by the user at the UDRDR register. The second configuration can be used at circular topography structure (see *Figure 576*). Standard transmission is re-enabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. The user must write some data into TxFIFO prior clearing UDR flag to prevent any next underrun condition occurrence capture.

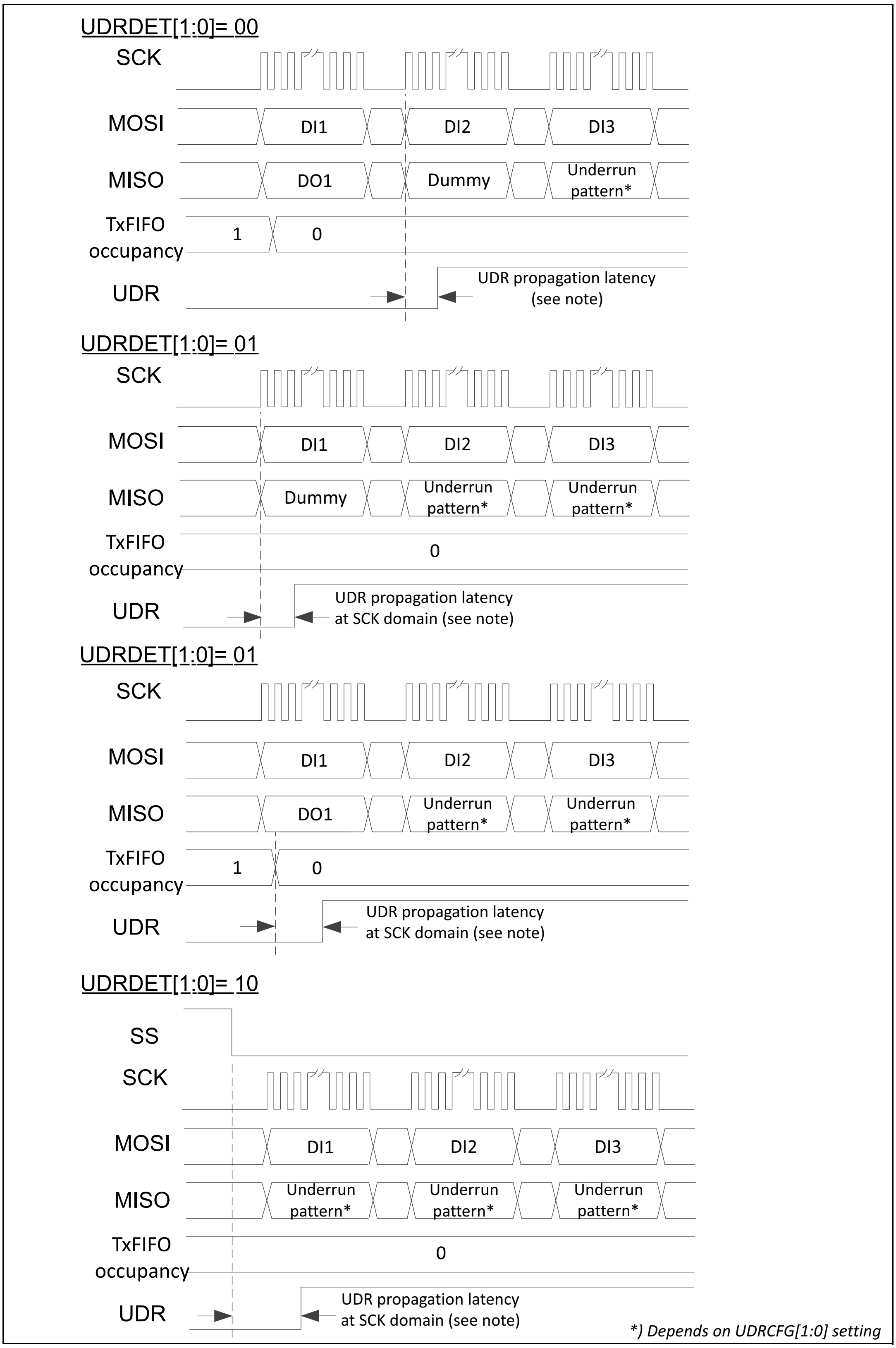
When the configuration UDRDET[1:0]=00 is applied, the underrun condition is evaluated whenever master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between last valid data and proper underrun pattern defined by UDRCFG[1:0]. This does not happen when any other UDRDET[1:0] configuration is applied suppose the slave's TxFIFO is not empty when underrun condition is checked (see *Figure 585: Optional configurations of slave behavior at detection of underrun condition*).

The data transacted by slave is unpredictable especially when the transaction starts or continues while TxFIFO is empty and underrun condition is either not yet captured or just cleared. Typically, this is the case when UDRDET[1:0]=00 or SPI is just enabled or when a transaction with a defined size just starts. First bits can be corrupted in this case, as well, when slave software writes first data into the empty TxFIFO too close prior the data transaction starts (propagation of the data into TxFIFO takes few APB clock cycles). If the user cannot ensure to write data into the empty TxFIFO in time the UDRDET[1:0]=00 setting must be avoided.

To handle the underrun control feature correctly the user must avoid next critical encroachments especially

- Any fill of empty TxFIFO when master starts clocking (at UDRDET[1:0]=00 especially)
- Any clear of UDR flag while TxFIFO is empty
- Any setting of UDRDET[1:0]=00 together with UDRCFG[1:0]=10 (to avoid repetition of undefined dummy data)
- Any setting of UDRDET[1:0]=10 when underrun must be detected after each data frame while SS signal does not toggle between the frames
- Any setting of UDRDET[1:0]=10 while SS is managed by software

Figure 585. Optional configurations of slave behavior at detection of underrun condition



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of data frame transaction or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by slave prior the valid data from TxFIFO becomes transacted again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal SS signal (SS pin in SS hardware mode, or SSI bit in SS software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the interrupt request is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero while MODF bit is set. This blocks all the peripheral outputs and disables the SPI interface.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit in the SPI_IFCR.

To avoid any multiple slave conflicts in a system comprising several MCUs, the SS pin must be pulled to its non-active level before re-enabling the SPI, by setting the SPE bit.

As a security, hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multi master conflict.

A correct SW procedure when master overtakes the bus at multi master system must be the following one:

- Switch into master mode while SSOE=0
  (potential conflict can appear when another master occupies the bus. MODF is raised in this case which prevents any next node switching into master mode)
- Put GPIO pin dedicated for another master SS control into active level
- Perform data transaction
- Put GPIO pin dedicated for another master SS control into non active level
- Switch back to slave mode

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CFG1 register is set. The CRCE flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if RCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit in the SPI_IFCR.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an SS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the SS pulse is ignored, and the SPI waits for the next SS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of few data bytes.

The TIFRE flag is cleared by writing 1 to the TIFREC bit in the SPI_IFCR. If the TIFREIE bit is set, an interrupt is generated on the SS error detection. As data consistency is no longer guaranteed, communication must be re-initiated by software between master and slave.

52.5.3 CRC computation

Two separate 33-bit or two separate 17-bit CRC calculators are implemented in order to check the reliability of transmitted and received data. The SPI offers any CRC polynomial length (*number of bits involved in the CRC computation)* from 5 to 33 bits when maximum data size is 32-bit and from 5 to 17 bits for the peripheral instances where maximum data size is limited to 16-bit.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string. For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1, write 0x11021 to the SPI_CRCPOLY register for a 32-bit instance, whereas to obtain the full size for a 16-bit instance, write 0x1021 with the CRC33_17 bit set.

The CRCSIZE field in the SPI_CFG1 then defines how many the most significant bits from CRC calculation registers are transacted and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size while its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC pattern size, else the bits unused at the calculation are transacted and expected all zero at the end of the CRC pattern if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit in the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCERR flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has to be sent or received in the SPI_DR register.

The length of the transfer has to be defined by TSIZE and TSER. When the desired number of data is transacted, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

TSIZE cannot be set to 0xFFFF value if CRC is enabled. A correct way of sending e.g. 65535 data with CRC is to set:

– TSIZE= 0xFFFE and TSER=1 when data packet is configured to keep one data respective
– TSIZE= 0xFFFC and TSER=3 when data packet keeps 4 data (to ensure the TSIZE value aligned with packet size when its extension is applied).

In transmission, the CRC computation is frozen during CRC transaction and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when desired number of data is transacted. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is completed, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. Software has to check the CRCERR flag in the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCERR flag by writing 1 to the CRCERRC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode in order to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI at SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

52.6 Low-power mode management

The SPI has advanced low-power mode functions allowing it to transfer properly data between the FIFOs and the serial interface even when the **spi_pclk** clock is disabled.

In master mode the **spi_ker_ck** kernel clock is needed in order to provide the timings of the serial interface.

In slave mode, the **spi_ker_ck** clock can be removed as well during the transfer of data between the FIFOs and the serial interface. In this mode the clock is provided by the external SPI device.

When the **spi_pclk** clock is gated, (and the **spi_ker_ck** clock as well if the SPI is in slave), the SPI provides a wakeup event signal (**spi_wkup**) if a specific action requiring the activation of the **spi_pclk** clock is needed, such as:

- To fill-up the TxFIFO,
- To empty the RxFIFO,
- Other signaling: end of transfer, errors...

The generation of **spi_ker_ck** and **spi_pclk** clock are controlled by the RCC block according to register settings and the processors modes. Refer to the RCC section for details.

The **spi_pclk** clock request stays pending till a flag with enabled interrupt is set. That is why it is important to service these pending requests and clear their flag as soon as possible at system sensitive to the low power consumption especially and the application must acknowledge all pending interrupts events before switching the SPI to low-power mode (i.e. removing **spi_pclk**).

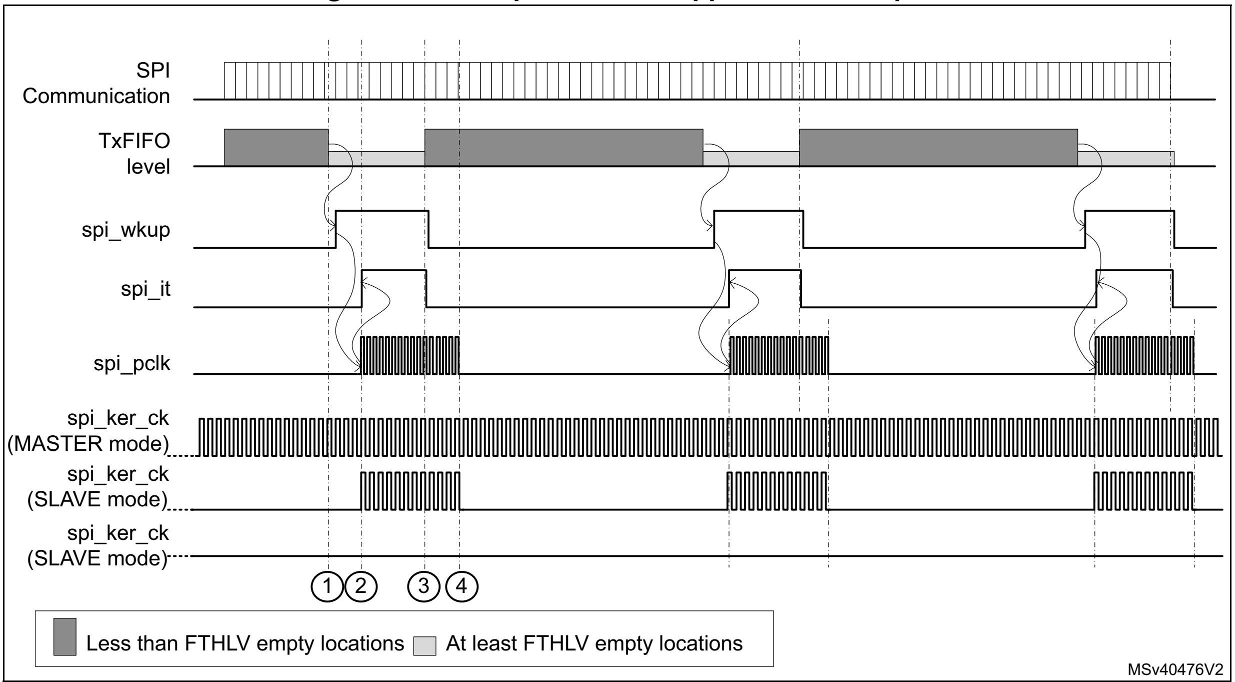
The *Figure 586* shows an example of the clock handling when the SPI2S is working in low-power mode. The example is given for a transmit mode.

In master mode the **spi_ker_ck** clock is required for the timing generation.

The *Figure 586* shows two kinds of supported scenarios for the handling of the **spi_ker_ck** kernel clock in slave mode:

- In most of the slave modes, the **spi_ker_ck** kernel clock can be disabled,
- In some products, the **spi_ker_ck** kernel clock activation may follow the system state.

Figure 586. Low-power mode application example



The figure clearly shows that the **spi_pclk** must be provided to the SPI2S, when data need to be transferred from the memory to the SPI2S TxFIFO. Here is the description of the most important steps:

- **Step 1**
  The TxFIFO level goes below the programmed threshold, this event (TXP) activates the **spi_wkup** signal. This signal is generally used to wake-up the system from low-power mode, and thus to activate the bus clock (**spi_pclk**).
- **Step 2**
  When **spi_pclk** is activated, the **spi_it** is also activated, and the product is ready to fill-up the TxFIFO either by DMA or by software. Note as well that for some product the system wake-up automatically enables the **spi_ker_ck** kernel clock as well.
- **Step 3**
  When the amount of empty locations in the TxFIFO is less than FTHLV, then the **spi_wkup** and **spi_it** signals are deactivated, but the fill-up of the TxFIFO may

continue. Note that **spi_wkup** falling edge is aligned with the serial interface clock domain, and the falling edge of the **spi_it** is aligned with the **spi_pclk** clock domain.

- **Step 4**
  The fill-up of the TxFIFO is completed; the software can switch the system back to low-power mode until the next **spi_wkup** occurs.

52.7 SPI wakeup and interrupts

Table 374 gives an overview of the SPI events capable to generate interrupt events (**spi_it**). Some of them feature wake-up from low-power mode capability additionally (**spi_wkup**).

Most of them can be enabled and disabled independently while using specific interrupt enable control bits.

The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. All the pending interrupt requests stay active if the SPI is disabled. A not cleared request with an enabled interrupt generates a spi_pclk clock request and so increases the overall consumption.

Table 374. SPI wakeup and interrupt requests

<table>
<tr><th rowspan="2">Interrupt event</th><th rowspan="2">Event flag(1)</th><th rowspan="2">Enable Control bit</th><th rowspan="2">Event clear method</th><th colspan="2">Interrupt/Wakeup activated</th></tr>
<tr><th>spi_it</th><th>spi_wkup</th></tr>
<tr><td>TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)</td><td>TXP</td><td>TXPIE</td><td>TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations</td><td rowspan="13">YES</td><td>YES</td></tr>
<tr><td>Data received in RxFIFO (one data packet available - FIFO threshold)</td><td>RXP</td><td>RXPIE</td><td>RXP cleared by hardware when RxFIFO contains less than FTHLV samples</td><td>YES</td></tr>
<tr><td>Both TXP and RXP active</td><td>DXP</td><td>DXPIE</td><td>When TXP or RXP are cleared</td><td>YES</td></tr>
<tr><td>Transmission Transfer Filled</td><td>TXTF</td><td>TXTFIE</td><td>Writing TXTFC to 1</td><td>NO</td></tr>
<tr><td>Underrun</td><td>UDR</td><td>UDRIE</td><td>Writing UDRC to 1</td><td>YES</td></tr>
<tr><td>Overrun</td><td>OVR</td><td>OVRIE</td><td>Writing OVRC to 1</td><td>YES</td></tr>
<tr><td>CRC Error</td><td>CRCE</td><td>CRCEIE</td><td>Writing CRCEC to 1</td><td>YES</td></tr>
<tr><td>TI Frame Format Error</td><td>TIFRE</td><td>TIFREIE</td><td>Writing TIFREC to 1</td><td>NO</td></tr>
<tr><td>Mode Fault</td><td>MODF</td><td>MODFIE</td><td>Writing MODFC to 1</td><td>NO</td></tr>
<tr><td>End Of Transfer (full transfer sequence completed - based on TSIZE value)</td><td>EOT</td><td rowspan="3">EOTIE</td><td>Writing EOTC to 1</td><td>YES</td></tr>
<tr><td>Master mode suspended</td><td>SUSP</td><td>Writing SUSPC to 1</td><td>YES</td></tr>
<tr><td>TxFIFO transmission complete (TxFIFO empty)</td><td>TXC(2)</td><td>TXC cleared by HW when a transmission activity starts on the bus</td><td>NO</td></tr>
<tr><td>TSER value transferred to TSIZE (new value may be loaded to TSER)</td><td>TSERF</td><td>TSERFIE</td><td>Writing TSERFC to 1</td><td>NO</td></tr>
</table>

1. Refer to SPI2S register description for more details about the event flags.
2. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

52.8 I2S main features

- Full duplex communication
- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length may be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, Overrun and Frame Error
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

a. 24- and 32-bit data width are not always available. Refer to *Section 52.3: SPI implementation*.

52.9 I2S functional description

52.9.1 I2S general description

The block diagram shown on *Figure 571* also applies for I2S mode.

The SPI/I2S block can work on I2S/PCM mode, when the bit I2SMOD is set to 1. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA and parts of interrupt signaling are shared with SPI function. The low-power mode function is also available in I2S mode, refer to *Section 52.6: Low-power mode management* and *Section 52.10: I2S wakeup and interrupts*.

52.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: Serial Data Output (mapped on the MOSI pin) to transmit the audio samples in master, and to receive the audio sample in slave. Refer to *Section : Serial Data Line swapping on page 2585*.
- SDI: Serial Data Input (mapped on the MISO pin) to receive the audio samples in master, and to transmit the audio sample in slave. Refer to *Section : Serial Data Line swapping on page 2585*.
- WS: Word Select (mapped on the SS pin) is the frame synchronization. It is configured as output in master mode, and as input for slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial bit clock. It is configured as output in master mode, and as input for slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: the Master Clock (mapped separately) is used when the I2S is configured in master mode.

52.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = ‘1’), some bitfields are no longer relevant, and must be forced to a specific value in order to guarantee the behavior of the I2S/PCM function. *Table 375* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 375. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI control register 2 (SPI_CR2)	-	Set to reset value
SPI configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S transmit data register (SPI_TXDR)	The complete register	-
SPI/I2S receive data register (SPI_RXDR)	The complete register	-
SPI polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI transmitter CRC register (SPI_TXCRC)	-	
SPI receiver CRC register (SPI_RXCRC)	-	
SPI underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

52.9.4 Slave and master modes

The SPI/I2S block supports master and slave mode for both I2S and PCM protocols.

In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

In order to improve the robustness of the SPI/I2S block in slave mode, the peripheral re-synchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.

Note: This re-synchronization mechanism is not available for the I2S LSB justified standard.

Note: Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

52.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel shall be considered as the left, and which one is the right.

In I2S master mode, four frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[a]
- 32-bit data packed in a 32-bit channel[a]

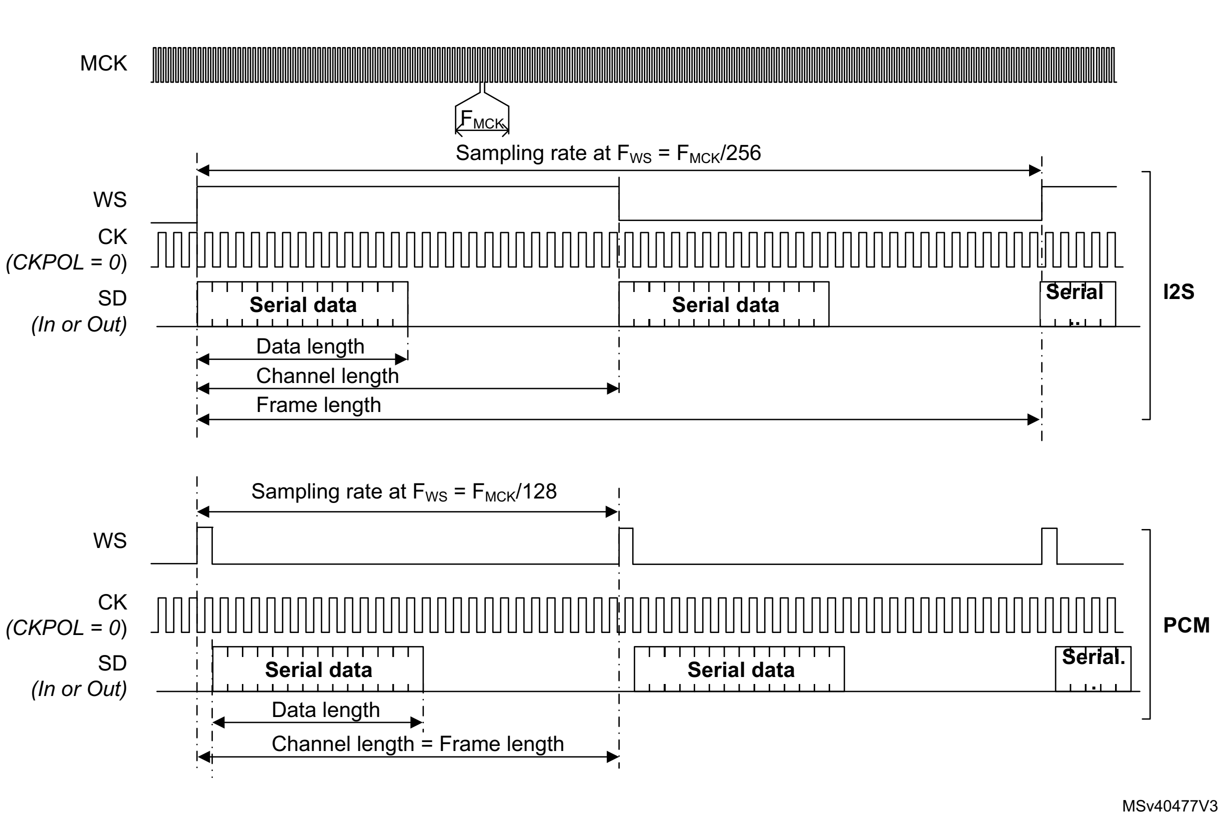
In PCM master mode, three frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[a]

a. 24- and 32-bit data width are not always available. Refer to *Section 52.3: SPI implementation*.

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 587. Waveform examples



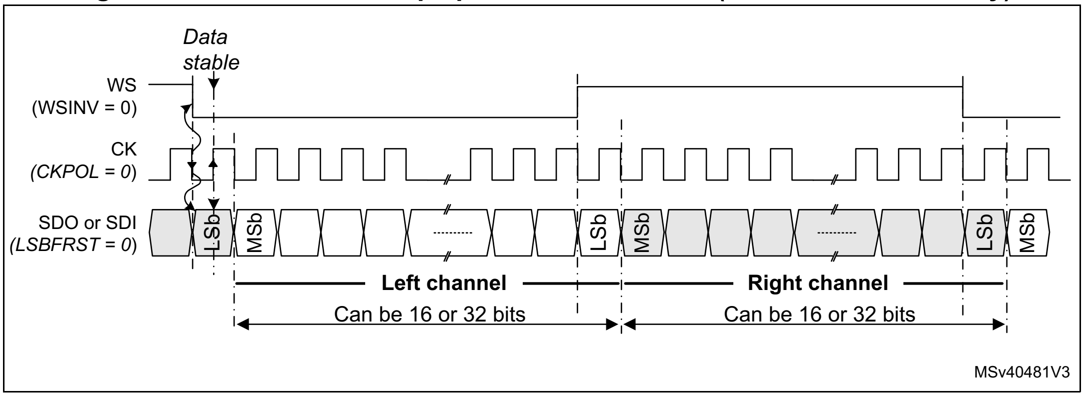
1. The channel length (CHLEN) must always be higher or equal to the data length (DATLEN).

I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transfered is the left channel, and a rising edge transition indicates that the next data transfered is the right channel.

Figure 588. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



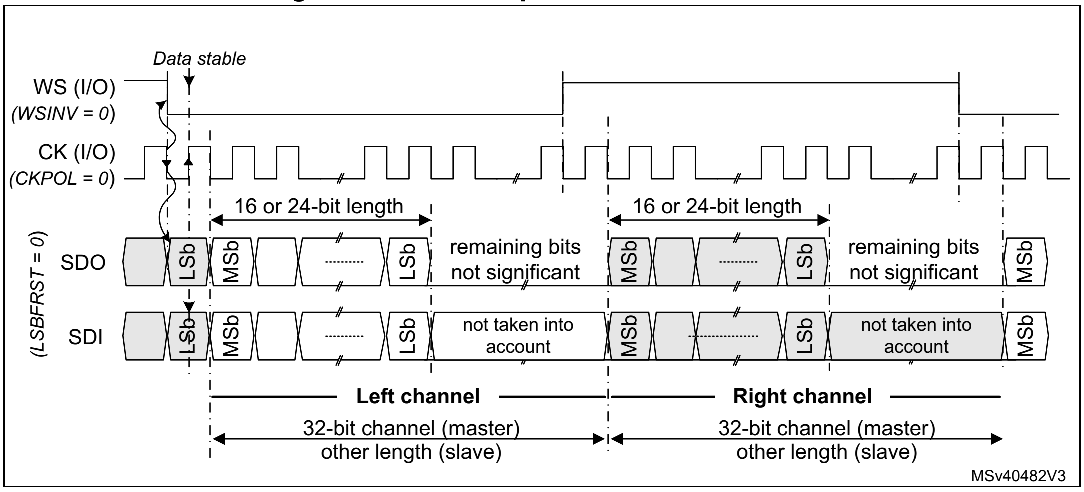
1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

CKPOL is set to 0 in order to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

Figure 588 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS Inversion* for information concerning the handling of WS signal.

Figure 589. I2S Philips standard waveforms



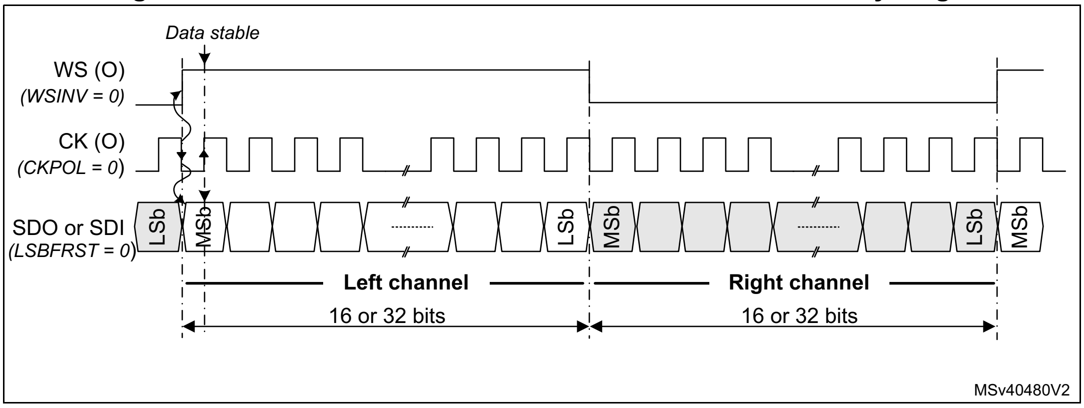
1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are forced to zero when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB justified standard

For this standard, the WS signal toggles when the first data bit, is provided. The data transfered represents the left channel if WS is high, and the right channel if WS is low.

Figure 590. Master MSB Justified 16-bit or 32-bit full-accuracy length

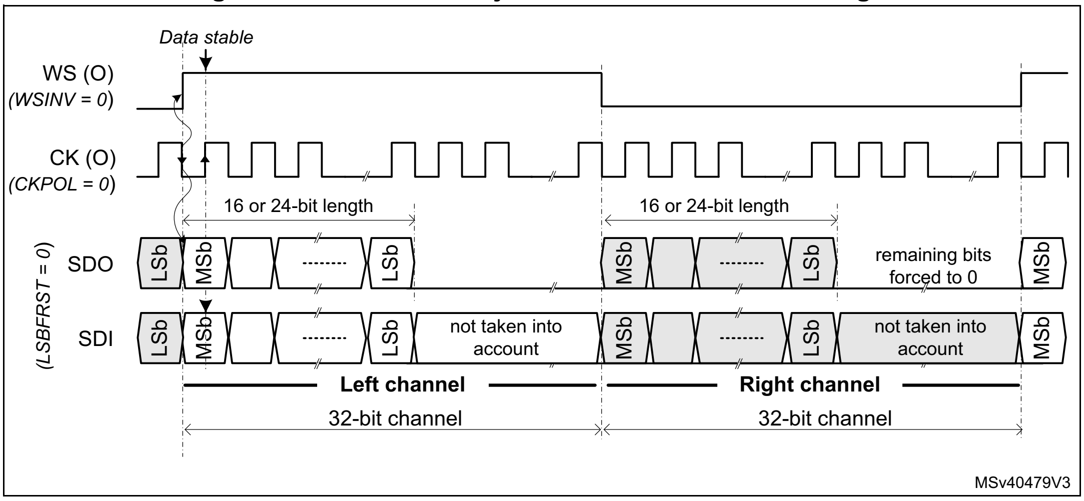


1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

CKPOL is set to 0 in order to match the I2S MSB justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

See *Control of the WS Inversion* for information concerning the handling of WS signal.

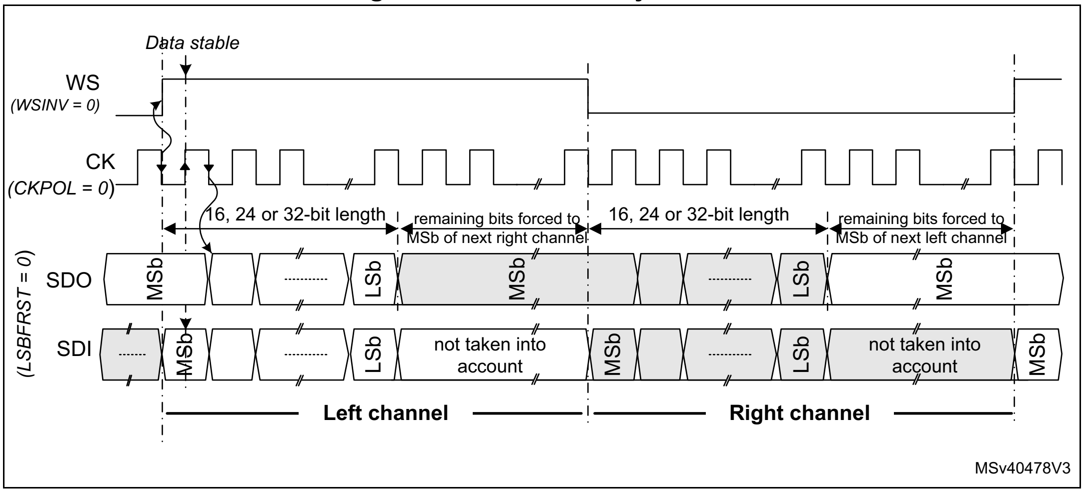
Figure 591. Master MSB justified 16 or 24-bit data length



1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are forced to zero when the SPI/I2S is configured in master transmit mode. In slave transmit the remaining bits are forced to the value of the first bit of the next data to be generated in order to avoid timing issues (see *Figure 592*).

Figure 592. Slave MSB justified



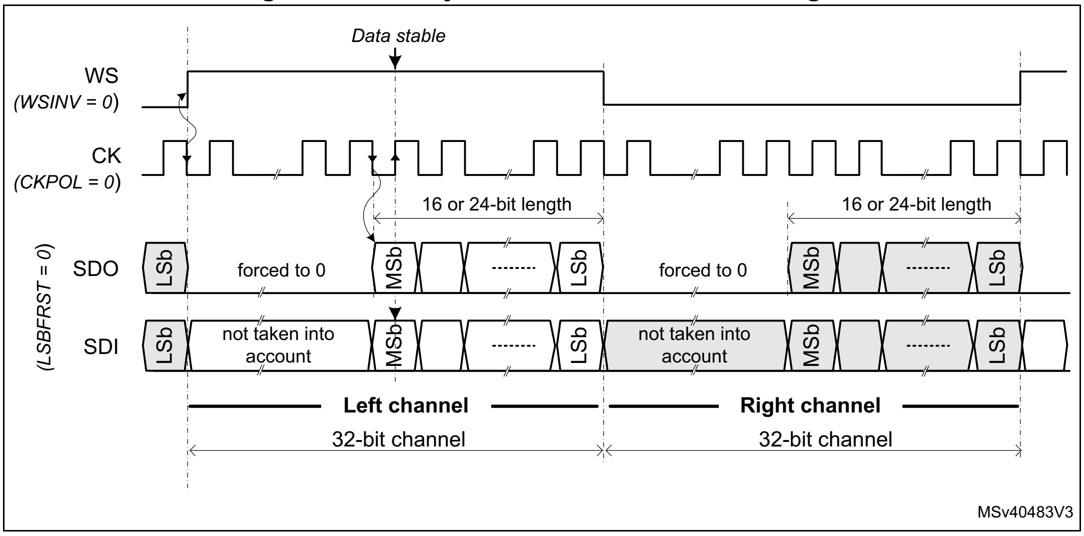
1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

LSB justified standard

This standard is similar to the MSB justified standard in master mode (no difference for the 16 and 32-bit full-accuracy frame formats). The LSB justified 16 or 32-bit full-accuracy format give similar waveforms than MSB justified mode (see *Figure 590*) because the channel and data have the same length.

Note: In the LSB justified format, only 16 and 32-bit channel length are supported in master and slave mode. This is due to the fact that it is not possible to transfer properly the data if the channel length is not known by transmitter and receiver side.

Figure 593. LSB justified 16 or 24-bit data length



1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

CKPOL is set to 0 in order to match the I2S LSB justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

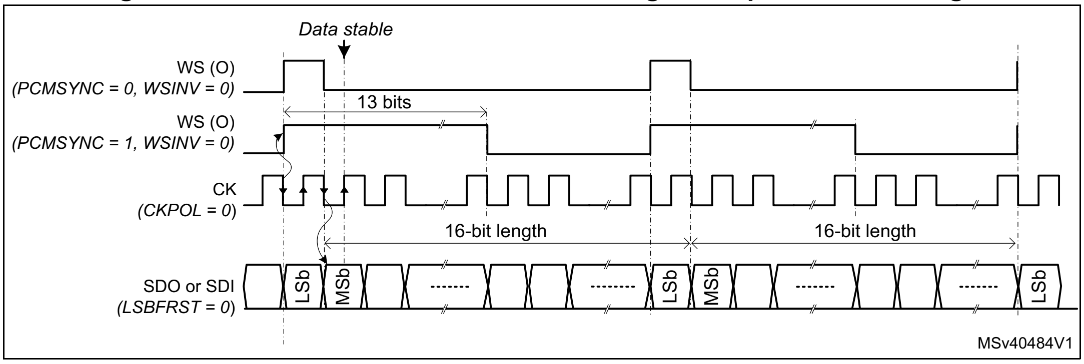
See *Control of the WS Inversion* for information concerning the handling of WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and configurable using the PCMSYNC bit in SPI_I2SCFGR register.

Note: The difference between the PCM long and short frame, is just the width of the frame synchronization: for both protocols, the active edge of the frame is generated (or is expected for the Slave mode) one CK clock cycle before the first bit.

Figure 594. Master PCM when the frame length is equal the data length



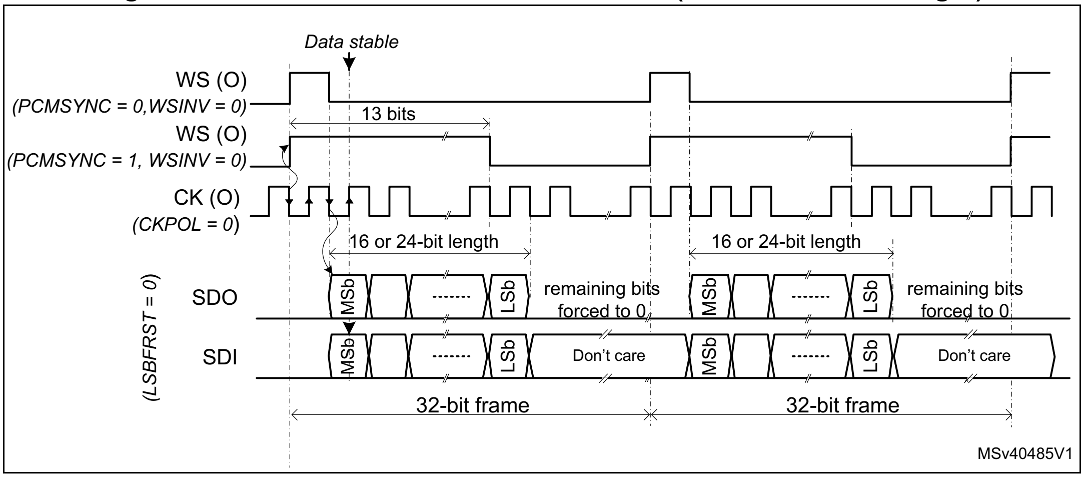
For long frame synchronization, the WS signal assertion time is fixed to 13 bits in master mode.

A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

For short frame synchronization, the WS synchronization signal is only one cycle long.

See *Control of the WS Inversion* for information concerning the handling of WS signal.

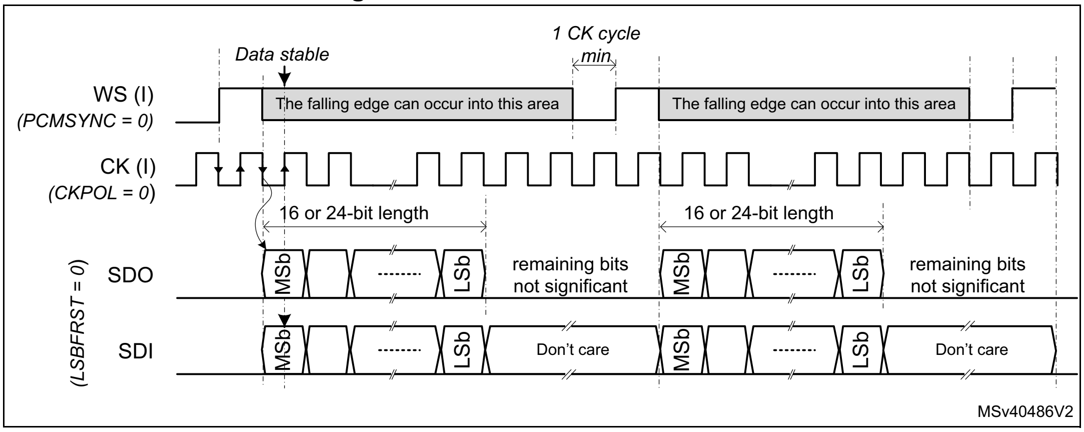
Figure 595. Master PCM standard waveforms (16 or 24-bit data length)



1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 596*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 596. Slave PCM waveforms



1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

CKPOL is set to 0 in order to match the PCM protocol. See *Selection of the CK sampling edge* for information concerning the handling of WS signal.

52.9.6 Additional Serial Interface Flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for I2S LSB justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS Inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV to 1. By default the WS polarity is the following:

- In I2S Philips Standard, WS is low for left channel, and high for right channel
- In MSB/LSB justified mode, WS is high for left channel, and low for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set to 1, the WS polarity is inverted, then:

- In I2S Philips Standard, WS is high for left channel, and low for right channel
- In MSB/LSB justified mode, WS is low for left channel, and high for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located into *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially set to 0

– Set I2SMOD = 1, (In order to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).

– Set bits CKPOL and WSINV to the wanted value.

– Set AFCNTR = 1.
Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.

– Then performs the activation sequence of the I2S/PCM

Table 376 shows the level of WS and CK signals, when the AFCNTR bit is set to 1, and before the SPI/I2S block is enabled (i.e. inactive level). Note that the level of WS depends also on the protocol selected.

Table 376. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: To avoid electrical conflicts on WS and CK, the AFCNTR bit must not be set to 1 when the SPI2S is in slave mode.

Serial Data Line swapping

The SPI/I2S offers the possibility to swap the function of SDI and SDO lines thanks to IOSWP bit located into *SPI configuration register 2 (SPI_CFG2)*. *Table 377* gives details on this feature.

Table 377. Serial data line swapping

Configuration	IOSWP	SDI direction	SDO direction
Master/slave RX	0	IN	-
	1	-	IN

Table 377. Serial data line swapping (continued)

Configuration	IOSWP	SDI direction	SDO direction
Master/slave TX	0	-	OUT
	1	OUT	-
Master/slave Full-duplex	0	IN	OUT
	1	OUT	IN

For simplification, the waveforms shown in the *I2S functional description* section have been done with IOSWP = 0.

52.9.7 Startup sequence

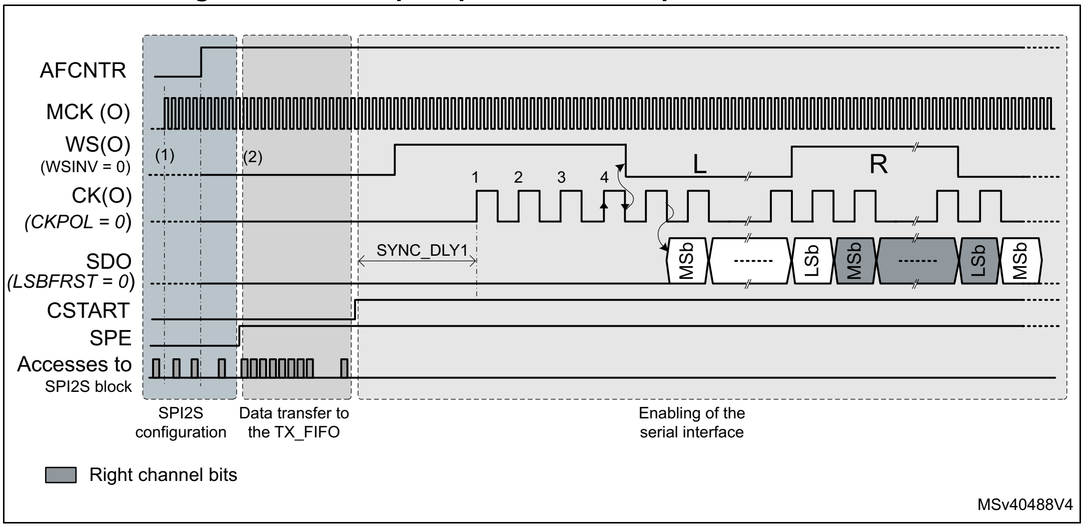
When the bit SPE is set to 0, the user is not allowed to read and write into the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

When the application wants to use the SPI/I2S block the user has to proceed as follow:

1. Ensure that the SPE is set to 0, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 52.9.16* for detailed programming examples.
3. Clear all the status flags by setting the SUSPC, TIFREC, OVRC and UDRC bits of SPI_IFCR register. Note that if the flag SUSP is not cleared (via SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to 1, in order to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO.
5. Set bit CSTART to 1, in order to activate the serial interface.

As shown in *Figure 597*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1 and the TxFIFO is not empty. Note that the bit clock CK is activated 4 rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 597. Startup sequence, I2S Philips standard, master



1. In this figure, the MCK is enabled before setting the bit SPE to 1. See *MCK generation* for more information.
2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set to 1

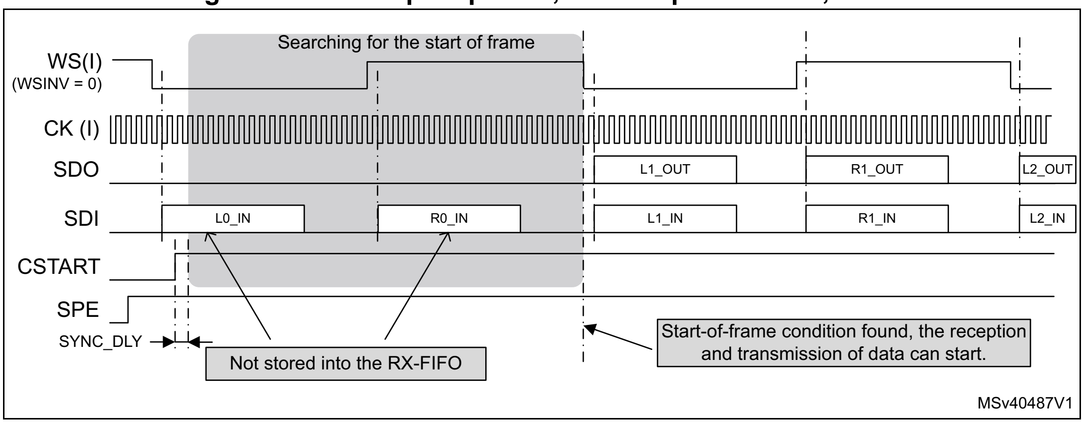
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about 3 periods of CK clock (SYNC_DLY1).

In slave mode, once the bit CSTART is set to 1, the data transfer starts when the start-of-frame condition is met:

- For I2S Philips standard, the start-of-frame condition is a falling edge of WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of WS signal. The transmission/reception starts at rising edge of WS for MSB aligned protocol.The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 598 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 598. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after 2 periods of CK clock (SYNC_DLY).

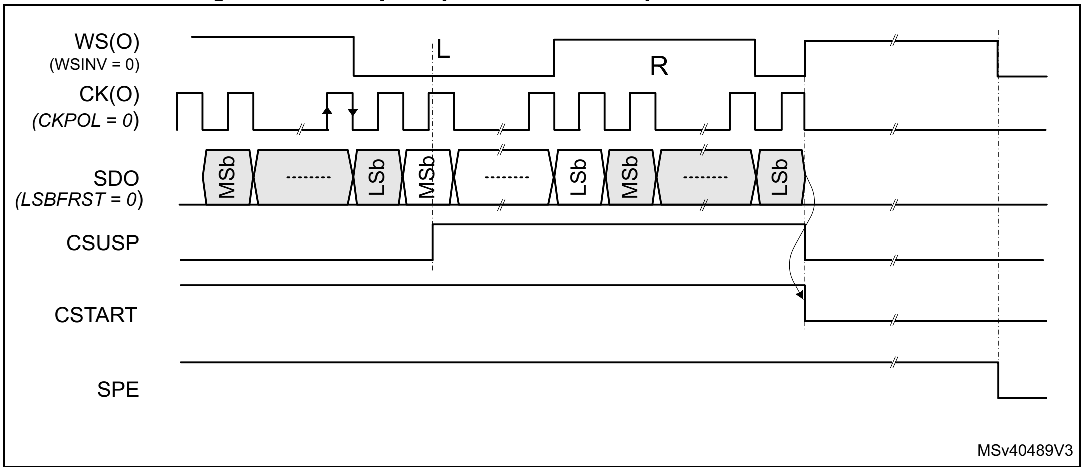
52.9.8 Stop sequence

The application can stop the I2S/PCM transfers by setting the SPE bit to 0. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP to 1, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be set to 0.

The *Figure 599* shows an example of stop sequence in the case of master mode. The CSUSP bit is set to 1, during the transmission of left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can set SPE to 0.

Figure 599. Stop sequence, I2S Philips standard, master



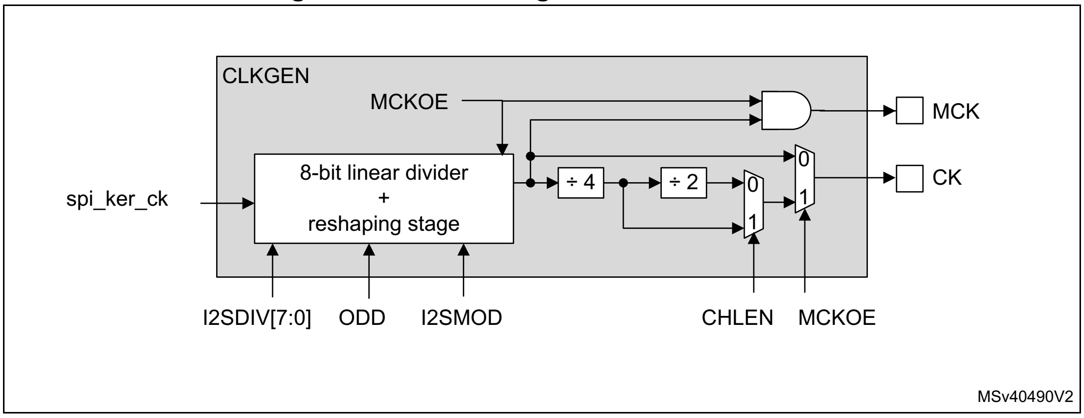
Note: In slave mode, the stop sequence is only controlled by the SPE bit.

52.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the Frame Synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.

If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 600. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 128 or 256 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there is no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

and by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s\_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s\_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.
I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: When {(2 x I2SDIV) + ODD} is odd, the duty cycle of MCK or the CK signals is not 50%. Care must be taken when odd ratio is used: it can impact margin on setup and hold time. For example if {(2 x I2SDIV) + ODD} = 5, then the duty cycle can be 40%.

Table 378 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK can be generated regardless to the SPE bit. The MCK generating is controlled by the following bits:

– I2SMOD must equal to 1,
– I2SCFG must select a master mode,
– MCKOE must be set to 1

Table 378. CLKGEN programming examples for usual I2S frequencies

<table>
<tr><th>i2s_clk (MHz)</th><th>Channel length (bits)</th><th>I2SDIV</th><th>ODD</th><th>MCK</th><th>Sampling rate: F_{WS} (kHz)</th></tr>
<tr><td>12.288</td><td>16</td><td>12</td><td>0</td><td rowspan="10">No</td><td>16</td></tr>
<tr><td>12.288</td><td>32</td><td>6</td><td>0</td><td>16</td></tr>
<tr><td>12.288</td><td>16</td><td>6</td><td>0</td><td>32</td></tr>
<tr><td>12.288</td><td>32</td><td>3</td><td>0</td><td>32</td></tr>
<tr><td>49.152</td><td>16</td><td>16</td><td>0</td><td>48</td></tr>
<tr><td>49.152</td><td>32</td><td>8</td><td>0</td><td>48</td></tr>
<tr><td>49.152</td><td>16</td><td>8</td><td>0</td><td>96</td></tr>
<tr><td>49.152</td><td>32</td><td>4</td><td>0</td><td>96</td></tr>
<tr><td>49.152</td><td>16</td><td>4</td><td>0</td><td>192</td></tr>
<tr><td>49.152</td><td>32</td><td>2</td><td>0</td><td>192</td></tr>
<tr><td>4.096</td><td>16 or 32</td><td>0</td><td>-</td><td rowspan="8">Yes</td><td rowspan="2">16</td></tr>
<tr><td>24.576</td><td>16 or 32</td><td>3</td><td>0</td></tr>
<tr><td>49.152</td><td>16 or 32</td><td>3</td><td>0</td><td>32</td></tr>
<tr><td>12.288</td><td>16 or 32</td><td>0</td><td>-</td><td rowspan="3">48</td></tr>
<tr><td>49.152</td><td>16 or 32</td><td>2</td><td>0</td></tr>
<tr><td>61.44</td><td>16 or 32</td><td>2</td><td>1</td></tr>
<tr><td>98.304</td><td>16 or 32</td><td>2</td><td>0</td><td>96</td></tr>
<tr><td>196.608</td><td>16 or 32</td><td>2</td><td>0</td><td>192</td></tr>
</table>

52.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

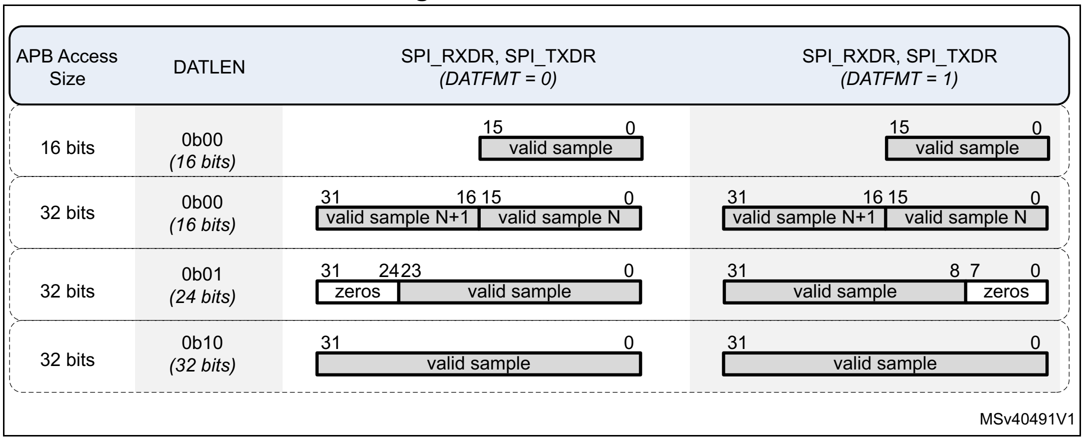
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 601 shows the allowed settings between APB access sizes, DATFMT and DATLEN.

Note: Caution shall be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 601. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
1. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is set to 0.

Refer to *Section 52.9.11* and *Section 52.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes 3 basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

52.9.11 FIFOs status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. thus FTHLV new data to be transmitted can be written into SPI_TXDR. The TXP flag is reset when the amount of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 52.10* for additional information on interrupt function in I2S mode.

52.9.12 Handling of underrun situation

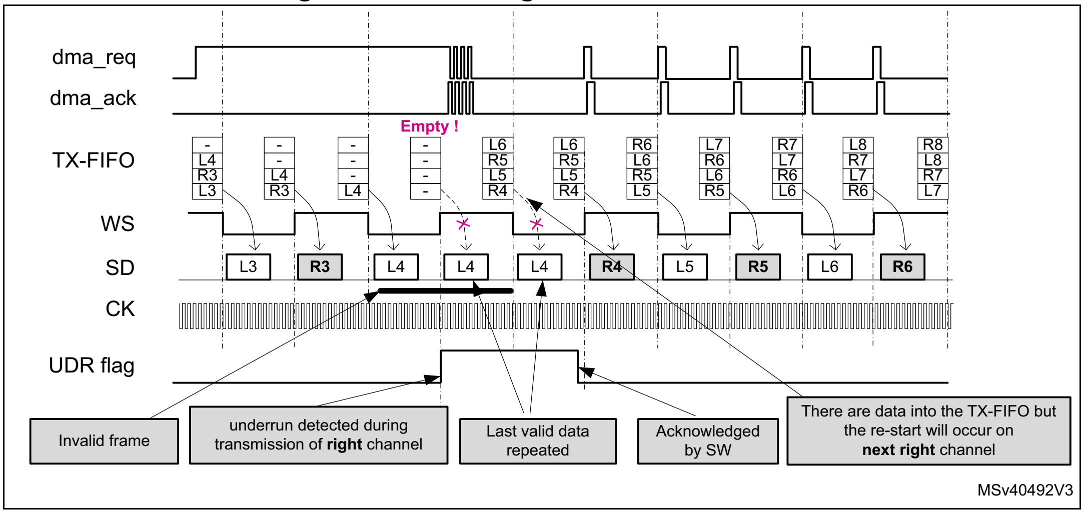
In transmit mode, the UDR flag is set when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation at least a data is lost.

In I2S mode, there is a hardware mechanism in order to prevent misalignment situation (left and right channel swapped). As shown in the following figure, when an underrun occurs, the peripheral re-plays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - if the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - if the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

a. 24- and 32-bit data width are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

Figure 602. Handling of underrun situation



The UDR flag can trigger an interrupt if the UDRIE bit in the SPI_IER register is set. The UDR bit is cleared by writing UDRC bit of SPI_IFCR register to 1.

When the block is configured in PCM mode, the transmission restarts at the start of the next frame when there is enough data into the TxFIFO, and the UDR flag is cleared by software.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK and MCK signal are not gated.

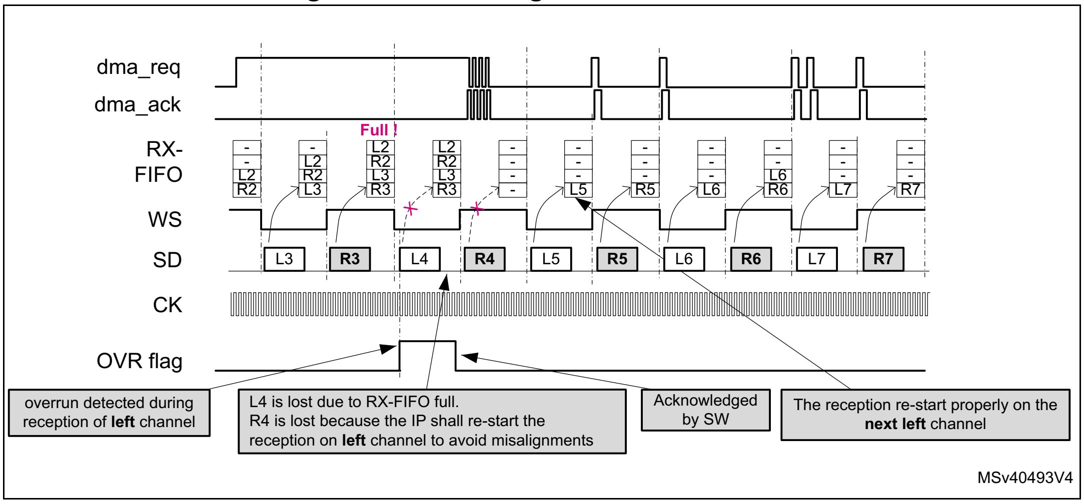
Due to resynchronization, any change on the UDR flag is taken into account by the hardware after at least 2 periods of CK clock.

52.9.13 Handling of overrun situation

The OVR flag is set when received data need to written into the RxFIFO, while the RxFIFO is already full. As a result, some incoming data are lost.

In I2S mode, there is a hardware mechanism in order to prevent misalignment situation (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as conditions of restart are not met. When there is enough room into the RxFIFO, and the OVR flag is cleared, the block starts by writing next the right channel into the RxFIFO if the overrun occurred when a right channel data was received or by writing the next left channel if the overrun occurred when a left channel data was received.

Figure 603. Handling of overrun situation



An interrupt may be generated if the OVRIE bit is set in the SPI_IER register. The OVR bit is cleared by writing OVRC bit of SPI_IFCR register to 1.

When the block is configured in PCM mode, after an overrun error, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met. When there is enough room in the RxFIFO, and the OVR flag is cleared, the next received data are written into the RxFIFO.

Note: An overrun situation can occur in master or slave mode. In master mode, when an overrun occurs, the WS, CK and MCK signal are not gated.

52.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

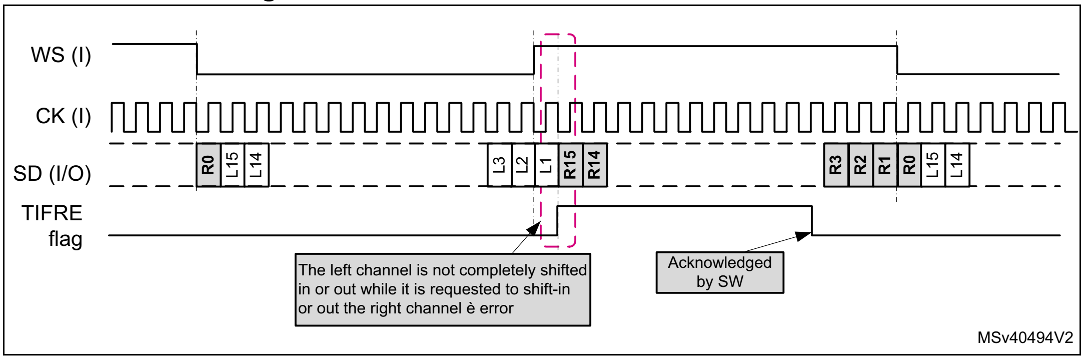
- A frame synchronization received while the shift-in or shift-out of the previous data is not completed (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurring at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to set FIXCH to 0. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH to 1 and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set to 1.

Figure 604 shows an example of frame error detection. The SPI/I2S block is in slave mode and the amount of bit clock periods for left channel are not enough to shift-in or shift-out the data. The figure shows that the on-going transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 604. Frame error detection, with FIXCH=0

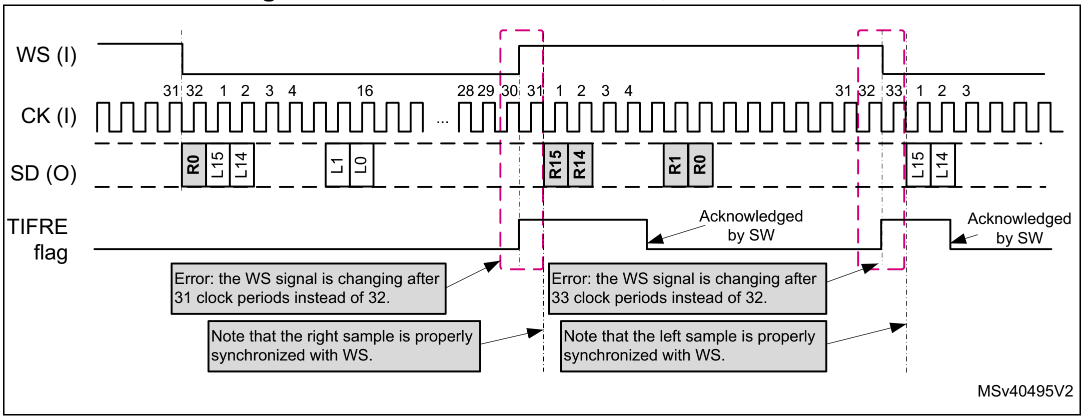


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH to 1. When this bit is set to 1, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 605*, in this mode the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 604* and *Figure 605* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 605. Frame error detection, with FIXCH=1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

52.9.15 DMA Interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of SPI_CFG1 register.

In receive mode, the DMA interface is working as follow:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follow:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

52.9.16 Programing examples

Master I2S Philips standard, transmit

This example shows how to program the interface for supporting the Philips I2S standard in master transmit mode, with a sampling rate of 48 kHz, using the master clock. The assumption taken is that SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit.

Start Procedure

1. Enable the bus interface clock (pclk or hclk), release the reset signal if needed in order to be able to program the SPI/I2S block.
2. Ensure that the SPI/I2S block receives properly a kernel frequency (at 61.44 MHz in this example).
3. Ensure that SPE is set to 0.
4. Program the clock generator in order to provide the MCK clock and to have a frame synchronization rate at exactly 48 kHz. So I2SDIV = 2, ODD = 1, and MCKOE = 1.
5. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (Philips Standard), I2SCFG = 2 (master transmit), I2SMOD = 1, for I2S/PCM mode. The register SPI_I2SCFGR must be updated before going to next steps.
6. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
7. Clear all status flag registers.
8. Enable the flags who shall generate an interrupt such as UDRIE. Note that TIFRE is not meaningful in master mode.
9. If the data transfer uses DMA:
   a) Program the DMA peripheral,
   b) Initialize the memory buffer with valid audio samples,
   c) Enable the DMA channel,
10. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the TXPIE bit to 1.
11. Set SPE to 1, as soon as this bit is set to one the following actions may happen:
    – If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
    – If the DMA transfer are enabled (TXDMAEN = 1), the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
12. Finally, the user has to ensure that the TxFIFO is not empty before enabling the serial interface. This is important in order to avoid an underrun condition when the interface is enabled. Then the SPI/I2S block can be enabled by setting the bit CSTART to 1. CSTART bit is located into SPI_CR1 register.

Stop Procedure in master mode

1. Set the bit CSUSP to 1, in order to stop on-going transfers
2. Check the value of CSTART bit until it goes to 0
3. Stop DMA peripheral, bus clock...
4. Set bit SPE to 0 in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So we cannot use the capability offered for frame error detection when FIXCH is set to 1.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI.
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLExN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags who shall generate an interrupt such as UDRIE and TIFRE.
7. If the data transfer uses DMA:
   - Program the DMA peripheral: one RX channel
   - Enable the DMA channel,
   - In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit to 1.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit to 1.
9. Set SPE to 1.
10. Finally the user can set the bit CSTART to 1 in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop Procedure in slave mode

1. Set bit SPE to 0 in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

52.10 I2S wakeup and interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wakeup event signal (**spi_wkup**) can be generated according to the events described in the *Table 379*.

Interrupt events can be enabled and disabled separately.

Table 379. I2S interrupt requests

<table>
<tr><th rowspan="2">Interrupt event</th><th rowspan="2">Event flag</th><th rowspan="2">Enable control bit</th><th rowspan="2">Event clear method</th><th colspan="2">Interrupt/Wakeup activated</th></tr>
<tr><th>spi_it</th><th>spi_wkup</th></tr>
<tr><td>TxFIFO threshold reached</td><td>TXP</td><td>TXPIE</td><td>TXP flag is cleared when the TxFIFO contains less than FTHLV empty locations</td><td rowspan="5">YES</td><td rowspan="4">YES</td></tr>
<tr><td>RxFIFO threshold reached</td><td>RXP</td><td>RXPIE</td><td>RXP flag is cleared when the RxFIFO contains less than FTHLV samples</td></tr>
<tr><td>Overrun error</td><td>OVR</td><td>OVRIE</td><td>OVR is cleared by writing OVRC to 1</td></tr>
<tr><td>Underrun error</td><td>UDR</td><td>UDRIE</td><td>UDR is cleared by writing UDRC to 1</td></tr>
<tr><td>Frame error flag</td><td>TIFRE</td><td>TIFREIE</td><td>TIFRE is cleared by writing TIFREC to 1</td><td>NO</td></tr>
</table>

52.11 SPI/I2S registers

52.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IO LOCK
															rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCRC INI	RCRC INI	CRC33 _17	SSI	HDDIR	CSUSP	C START	MAS RX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit is set by software and cleared by hardware whenever SPE bit is changed from 1 to 0.

0: AF configuration is not locked

1: AF configuration is locked

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit can be set when SPI is disabled only else it is write protected. It is cleared and cannot be set when the MODF bit is set.

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: All zero pattern is applied

1: All ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: all zero pattern is applied

1: all ones pattern is applied

Bit 13 **CRC33_17**: 32-bit CRC polynomial configuration

0: full size (33-bit or 17-bit) CRC polynomial is not used

1: full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** internal SS signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral SS input and the I/O value of the SS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at Half-duplex mode

In Half-Duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is Receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transaction.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before disabling the SPI or going to low-power mode. This bit can be used in SPI or I2S mode.

After software suspension, the SUSP flag has to be cleared and the SPI disabled and re-enabled before any next transaction starts.

Bit 9 **CSTART:** master transfer start

This bit is set by software to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when End Of Transfer (EOT) flag is set or when a transaction suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the section stop sequence.

0: master transfer is at idle

1: master transfer is on-going or temporary suspended by automatic suspend

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

Bit 8 **MASRX:** master automatic SUSP in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when SPI communication is suspended.

When SPI communication is suspended by hardware automatically, it could happen that few bits of next frame are already clocked out due to internal synchronization delay.

That is why the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transaction without any next constraint. Prior the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

0: serial peripheral disabled.

1: serial peripheral enabled

When SPE=1, the SPI data transfer is enabled, the configuration registers SPI_CFG1, SPI_CFG2, CRCPOLY and UDRDR and the IOLOCK bit in SPI_CR1 are write protected. They can be changed only when SPE=0.

When SPE=0 any SPI operation is stopped and disabled, all the not cleared requests with enabled interrupt stay pending and propagates the spi_plck clock request, the SS output is deactivated at master, internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE is cleared and cannot be set when MODF error flag is active.

52.11.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSER[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TSER[15:0]**: number of data transfer extension to be reload into TSIZE just when a previous number of data stored at TSIZE is transacted

This register can be set by software when its content is cleared only. It is cleared by hardware once TSIZE reload is done. The TSER value must be programmed in advance before CTSIZE counter reaches zero otherwise the reload is not taken into account and traffic terminates with normal EOT event.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled. The field can be updated by hardware optionally, too, to be reloaded by the TSER value if applicable.

Endless transaction is initialized when CSTART is set while zero value is stored at TSIZE. TSIZE cannot be set to 0xFFFF value when CRC is enabled.

52.11.3 SPI configuration register 1 (SPI_CFG1)

Address offset: 0x08

Reset value: 0x0007 0007

Content of this register is write protected when SPI is enabled, except TXDMAEN and RXDMAEN bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXDMAEN	RXDMAEN	Res.	UDRDET[1:0]		UDRCFG[1:0]		FTHLV[3:0]				DSIZE[4:0]				
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **MBR[2:0]:** master baud rate
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256

Note: MBR setting is considered at slave working at TI mode, too (see Section 52.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation Enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transacted and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transacted or compared. The length of the polynomial is not affected by this setting.
00000: reserved
00001: reserved
00010: reserved
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits
The value must be set equal or multiply of data size (DSIZE[4:0]). Its maximum size cannot exceed the DSIZE maximum at the instance.

Note: If CRC calculation is disabled by CRCEN=0, the CRCSIZE field must be kept at its default setting.

Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where the data size is limited to 16-bit.

Bit 15 **TXDMAEN:** Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **UDRDET[1:0]:** detection of underrun condition at slave transmitter
00: underrun is detected at begin of data frame (no protection of 1-st bit)
01: underrun is detected at end of last data frame
10: underrun is detected by begin of active SS signal
11: reserved
For more details see *Figure 585: Optional configurations of slave behavior at detection of underrun condition*.

Bits 10:9 **UDRCFG[1:0]:** behavior of slave transmitter at underrun condition
00: slave sends a constant pattern defined by the user at SPI_UDRDR register
01: slave repeats lastly received data frame from master
10: slave repeats its lastly transmitted data frame
11: reserved
When slave is configured at transmit only mode (COMM[1:0]=01), slave repeats all zeros pattern at UDRCFG[1:0]=01 setting.
For more details see *Figure 585: Optional configurations of slave behavior at detection of underrun condition*.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level
Defines number of data frames at single data packet. The size of the packet must not exceed 1/2 of FIFO space.
0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data
SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:
– If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV=2, 4, 6 ..,
– If SPI data register is accessed as a 32-bit register and DSIZE>8bit, better to select FTHLV=2, 4, 6 .., while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12 ..

Bits 4:0 **DSIZE[4:0]:** number of bits in at single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4-bits
00100: 5-bits
00101: 6-bits
00110: 7-bits
00111: 8-bits
.....
11101: 30-bits
11110: 31-bits
11111: 32-bits

Note: The most significant bit at DSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.

52.11.4 SPI configuration register 2 (SPI_CFG2)

Address offset: 0x0C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled or IOLOCK bit is set at SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AF CNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSB FRST	MAS TER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IOSWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE=0 only

0: the peripheral takes no control of GPIOs while it is disabled
1: the peripheral keeps always control of all associated GPIOs
When SPI master has to be disabled temporary for a specific configuration reason (e.g. CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration. This bit must be never used at slave mode as any slave transmitter must not force its MISO output once the SPI is disabled.

Note: This bit can be also used in PCM and I2S modes.

Bit 30 **SSOM**: SS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows to configure SS output between two consecutive data transfers.

0: SS is kept at active level till data transfer is completed, it becomes inactive with EOT flag
1: SPI data frames are interleaved with SS non active pulses when MIDI[3:0]>1

Bit 29 **SSOE:** SS output enable

This bit is taken into account at master mode only

0: SS output is disabled and the SPI can work in multimaster configuration

1: SS output is enabled. The SPI cannot work in a multimaster environment. It forces the SS pin at inactive level after the transfer is completed or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP:** SS input/output polarity

0: low level is active for SS signal

1: high level is active for SS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** software management of SS signal input

0: SS input value is determined by the SS PAD

1: SS input value is determined by the SSI bit

When master uses hardware SS output (SSM=0 and SSOE=1), the SS signal input is forced to non active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity

0: SCK signal is at 0 when idle

1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase

0: the first clock transition is the first data capture edge

1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format

0: MSB transmitted first

1: LSB transmitted first

Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master

0: SPI Slave

1: SPI Master

Bits 21:19 **SP[2:0]:** Serial protocol

000: SPI Motorola

001: SPI TI

others: Reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI communication mode

00: full-duplex

01: simplex transmitter

10: simplex receiver

11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins

0: no swap

1: MOSI and MISO are swapped

When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.

Note: This bit can be also used in PCM and I2S modes.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: master SS idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS and first data of a session start in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.

52.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	TSERF IE	MODF IE	TIFRE IE	CRCE IE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
					rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **TSERFIE**: additional number of transactions reload interrupt enable

0: TSERF interrupt disabled

1: TSERF interrupt enabled

Bit 9 **MODFIE**: mode fault interrupt enable

0: MODF interrupt disabled

1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable

0: TIFRE interrupt disabled

1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable

0: CRC interrupt disabled

1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable

0: OVR interrupt disabled

1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTFIE interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP Interrupt Enable
0: RXP interrupt disabled
1: RXP interrupt enabled

52.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x14

Reset value: 0x0000 1002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXWNE	RXPLVL[1:0]		TXC	SUSP	TSERF	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus and at LP mode too.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data

Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE=0 and data size is set up to 16-bit, the value gives number of remaining data frames persisting at RxFIFO.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

(*) optional count when the data size is set up to 8-bit.

When the frame size is greater than 16-bit, these bits read as 00. In consequence, the single data frame received at the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if data size is set from 17 to 24 bits. The user then must apply other methods like TSIZE>0 or FTHLV=0.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE=0 the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE <>0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

0: Current data transaction is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: Last TxFIFO frame transmission completed

Bit 11 **SUSP:** suspension status

In Master mode, SUSP is set by hardware when a CSUSP request is done, either as soon as the current frame is completed after CSUSP request is done or at master automatic suspend receive mode (MASRX bit is set at SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit has to be cleared prior SPI is disabled by write 1 to SUSPC bit at SPI_IFCR

0: SPI not suspended (master mode active or other mode).

1: Master mode is suspended (current frame completed)

Bit 10 **TSERF**: additional number of SPI data to be transacted was reload

This bit is cleared by write 1 to TSERFC bit at SPI_IFCR or by writing the TSER[15:0] (SPI_CR2) register

0: no acceptation

1: additional number of data accepted, current transaction continues

Bit 9 **MODF**: mode fault

0: no mode fault

1: mode fault detected

This bit is cleared by write 1 to MODFC bit at SPI_IFCR. When MODF is set, the SPE and IOLOCK bits at the SPI_CR1 register are reset and their setting is blocked.

Bit 8 **TIFRE**: TI frame format error

0: no TI Frame Error

1: TI Frame Error detected

This bit is cleared by write 1 to TIFREC bit at SPI_IFCR

Bit 7 **CRCE:** CRC error

0: no CRC error

1: CRC error detected

This bit is cleared by write 1 to CRCEC bit at SPI_IFCR

Bit 6 **OVR:** overrun
0: no overrun
1: overrun detected
This bit is cleared by write 1 to OVRC bit at SPI_IFCR

Bit 5 **UDR:** underrun
0: no underrun
1: underrun detected
This bit is cleared by write 1 to UDRC bit at SPI_IFCR

Note: In SPI mode, the UDR flag applies to Slave mode only. In I2S/PCM mode, (when available) this flag applies to Master and Slave mode.

Bit 4 **TXTF:** transmission transfer filled
0: upload of TxFIFO is on-going or not started
1: TxFIFO upload is finished
TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.
This bit is cleared by software write 1 to TXTFC bit at SPI_IFCR
TXTF flag triggers an interrupt if TXTFIE bit is set.
TXTF setting clears the TXIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

Bit 3 **EOT:** end of transfer
EOT is set by hardware as soon as a full transfer is complete, that is when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared by software write 1 to EOTC bit at SPI_IFCR.
EOT flag triggers an interrupt if EOTIE bit is set.
If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.
0: transfer is on-going or not started
1: transfer complete
In master, EOT event terminates the data transaction and handles SS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transaction.

Bit 2 **DXP**: duplex packet
0: TxFIFO is Full and/or RxFIFO is Empty
1: Both TxFIFO has space for write and RxFIFO contains for read a single packet at least
The DXP flag is set whenever both TXP and RXP flags are set regardless the SPI mode.

Bit 1 **TXP:** Tx-packet space available
0: there is not enough space to locate next data packet at TxFIFO
1: TxFIFO has enough free location to host 1 data packet
TXP flag is changed by hardware. It monitors overall space currently available at TxFIFO if SPI is enabled. It has to be checked once a complete data packet is stored at TxFIFO.

Bit 0 **RXP:** Rx-packet available
0: RxFIFO is empty or a not complete data packet is received
1: RxFIFO contains at least 1 data packet
RXP flag is changed by hardware. It monitors number of overall data currently available at RxFIFO if SPI is enabled. It has to be checked once a data packet is completely read out from RxFIFO.

52.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	TSERFC	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w	w	w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** SUSPend flag clear

Writing a 1 into this bit clears SUSP flag in the SPI_SR register

Bit 10 **TSERFC**: TSERFC flag clear

Writing a 1 into this bit clears TSERF flag in the SPI_SR register

Note: *TSERF is also reset by writing the TSER[15:0] (SPI_CR2) register*

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag in the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag in the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag in the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag in the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag in the SPI_SR register

Bit 4 **TXTFC:** transmission Transfer Filled flag clear

Writing a 1 into this bit clears TXTF flag in the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag in the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

52.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: data is always right-aligned. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

52.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: Data is always right-aligned. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access.

halfword-wise (16 bit access) in this case 2 data-bytes or 1 halfword-data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 halfword-data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

52.11.10 SPI polynomial register (SPI_CRCPOLY)

Address offset: 0x40

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used at some other ST products with fixed length of the polynomial string where the most significant bit of the string is always kept hidden.

The length of the polynomial is given by the most significant bit of the value stored at this register. It has to be set greater than DSIZE. CRC33_17 bit has to be set additionally with CRCPOLY register when DSIZE is configured to maximum 32-bit or 16-bit size and CRC is enabled (to keep polynomial length greater than data size).

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

52.11.11 SPI transmitter CRC register (SPI_TXCRC)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: A read to this register when the communication is ongoing could return an incorrect value.

Not used for the I²S mode.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits at this case.

52.11.12 SPI receiver CRC register (SPI_RXCRC)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is written to 1 or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

Note: A read to this register when the communication is ongoing could return an incorrect value.

Not used for the I²S mode.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits at this case.

52.11.13 SPI underrun data register (SPI_UDRDR)

Address offset: 0x4C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account at slave mode and at underrun condition only. The number of bits considered depends on DSIZE bit settings at SPI_CFG1 register. Underrun condition handling depends on setting if UDRDET and UDRCFG bits at SPI_CFG1 register.

Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

52.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x50

Reset value: 0x0000 0000

Note: All documented bits in this register must be configured when the I2S is disabled (SPE = 0). These bits are not used in SPI mode except for I2SMOD which needs to be kept at 0 in SPI mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Refer to *Section 52.9.9: Clock generator* for details

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler

I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to *Section 52.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format

0: the data inside the SPI_RXDR or SPI_TXDR are right aligned
1: the data inside the SPI_RXDR or SPI_TXDR are left aligned.

Bit 13 **WSINV**: Word select inversion

This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB or LSB justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM mode, the data transfer starts one CK cycle after the rising edge of WS.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB or LSB justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM mode, the data transfer starts one CK cycle after the rising edge of WS.

Bit 12 **FIXCH**: fixed channel length in slave

0: the channel length in slave mode is different from 16 or 32 bits (CHLEN not taken into account)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity

0: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (i.e. SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (i.e. SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)

0: 16-bit wide
1: 32-bit wide
The selected channel length must always be higher or equal to the data length (DATLEN).

Bits 9:8 **DATLEN[1:0]**: data length to be transferred

00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: not allowed
The selected data length must never be higher than the channel length (CHLEN).

Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 52.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization

0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I^{2}S standard selection
00: I^{2}S Philips standard.
01: MSB justified standard (left justified)
10: LSB justified standard (right justified)
11: PCM standard
For more details on I^{2}S standards, refer to *Section 52.9.5: Supported audio protocols*

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - full duplex
101: master - full duplex
others, not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

52.11.15 SPI/I2S hardware configuration register (SPI_I2S_HWCFGR)

Address offset: 0x03F0

Reset value: 0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSCFG[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2SCFG[3:0]				CRCCFG[3:0]				RXFCFG[3:0]				TXFCFG[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DSCFG[3:0]:** SPI data size configuration
0000: data size is configurable from 4-bits to 16-bits
0001: data size is configurable from 4-bits to 32-bits
All the other combinations are reserved.

Bits 15:12 **I2SCFG[3:0]:** I2S configuration
0000: I2S support is not implemented
0001: I2S support is implemented
All the other combinations are reserved.

Bits 11:8 **CRCCFG[3:0]:** CRC configuration for SPI
0000: CRC support is not implemented
0001: CRC support is implemented
All the other combinations are reserved.

Bits 7:4 **RXFCFG[3:0]:** RxFIFO size
0002: the FIFO size is 4 bytes
0003: the FIFO size is 8 bytes
0004: the FIFO size is 16 bytes
0005: the FIFO size is 32 bytes
All the other combinations are reserved.

Bits 3:0 **TXFCFG[3:0]:** TxFIFO size
0002: the FIFO size is 4 bytes
0003: the FIFO size is 8 bytes
0004: the FIFO size is 16 bytes
0005: the FIFO size is 32 bytes
All the other combinations are reserved.

52.11.16 SPI/I2S version register (SPI_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** major revision of the IP.

Bits 3:0 **MINREV[3:0]:** minor revision of the IP.

52.11.17 SPI/I2S identification register (SPI_IPIDR)

Address offset: 0x03F8

Reset value: 0x0013 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP identification
MCD identification code which identifies the IP.

52.11.18 SPI/I2S size identification register (SPI_SIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: size identification
Bits[31:8]: fixed code 0xA3C5DD0
Bits[7:0]: range of address decoding boundary
0x01: 1 Kbyte
0x02: 2 Kbytes
0x04: 4 Kbytes
0x08: 8 Kbytes
All the other combination are reserved.

52.12 SPI register map and reset values

Table 380. SPI register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x04	SPI_CR2	TSER[15:0]																TSIZE[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	SPI_CFG1	Res.	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]					TXDMAEN	RXDMAEN	Res.	UDRDET [1:0]		UDRCFG [1:0]		FTHLV[3:0]				DSIZE[4:0]				
	Reset value		0	0	0						0		1	1	1	1	1	0	0		0	0	0	0	0	0	0	0	0	0	1	1	1
0x0C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM [1:0]		Res.	IOSWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0								0	0	0	0	0	0	0	0
0x10	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSERFIE	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x14	SPI_SR	CTSIZE[15:0]																RXWNE	RXPLVL [1:0]		TXC	SUSP	TSERF	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
0x18	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	TSERFC	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0	0	0	0	0	0	0	0	0			
0x20	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24 - 0x2C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x30	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34 - 0x3C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x40	SPI_CRCPOLY	CRCPOLY[31:16](1)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x44	SPI_TXCRC	TXCRC[31:16](1)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	SPI_RXCRC	RXCRC[31:16](1)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	SPI_UDRDR	UDRDR[31:16](1)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 380. SPI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x50	**SPI_I2SCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0		0	0	0	0	0	0
0x03F0	**SPI_I2S_ HWCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSCFG[3:0]				I2SCFG[3:0]				CRCCFG [3:0]				RXFCFG [3:0]				TXFCFG [3:0]			
	Reset value													0	0	0	x	0	0	0	x	0	0	0	x	0	x	x	x	0	x	x	x
0x03F4	**SPI_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0x03F8	**SPI_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
0x03FC	**SPI_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

1. Bits 31-16 are reserved at the peripheral instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored. *Refer to Table register boundary addresses.*

Refer to *Section 2.5 on page 153* for the register boundary addresses.

53 Serial audio interface (SAI)

53.1 Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

53.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Possible synchronization between multiple SAIs
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception

- Interrupt sources when enabled:
  - Errors
  - FIFO requests
- 2-channel DMA interface

53.3 SAI implementation

Table 381. STM32MP153 SAI features (1)

SAI features	SAI1	SAI2	SAI3	SAI4
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97	X	X	X	X
FIFO size	8 words	8 words	8 words	8 words
SPDIF	X	X	X	X
PDM	X (2)	-	-	X (2)

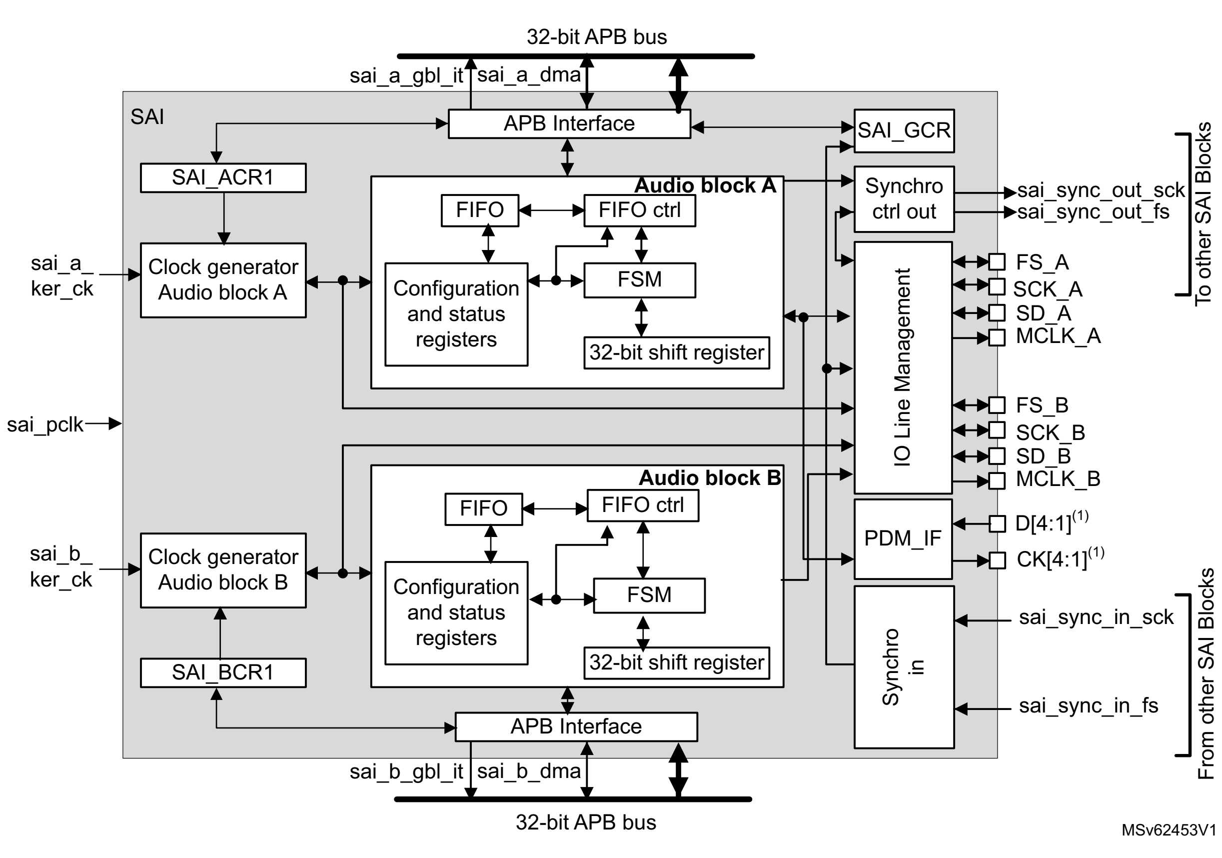
1. ‘X’ = supported, ‘-’ = not supported.
2. Only signals D[4:1], and CK[2:1] are available.

53.4 SAI functional description

53.4.1 SAI block diagram

Figure 606 shows the SAI block diagram while *Table 382* and *Table 383* list SAI internal and external signals.

Figure 606. SAI functional block diagram



1. These signals might not be available for all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC’97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where ‘x’ represents the SAI A or B subblock.

53.4.2 SAI pins and internal signals

Table 382. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 383. SAI input/output pins

Pin name	Pin type	Comments
SAI_SCK_A/B	Input/output	Audio block A/B bit clock
SAI_MCLK_A/B	Output	Audio block A/B master clock
SAI_SD_A/B	Input/output	Data line for block A/B
SAI_FS_A/B	Input/output	Frame synchronization line for audio block A/B
SAI_CK[4:1]	Output	PDM bitstream clock(1)
SAI_D[4:1]	Input	PDM bitstream data(1)

1. These signals might not be available in all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

53.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC’97 protocols.

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

53.4.4 SAI synchronization mode

There are two levels of synchronization, either at audio subblock level or at SAI level.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

External synchronization

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follows:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblocks is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

Note: The SYNCIN[1:0] and SYNCOUT[1:0] bits are located in the SAI_GCR register, and the SYNCEN bits in the SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as 'Reserved' must not be used.

Table 384. External synchronization selection

Block instance	SYNCIN= 3	SYNCIN= 2	SYNCIN= 1	SYNCIN= 0
SAI1	SAI4	SAI3	SAI2	Reserved
SAI2	SAI4	SAI3	Reserved	SAI1
SAI3	SAI4	Reserved	SAI2	SAI1
SAI4	Reserved	SAI3	SAI2	SAI1

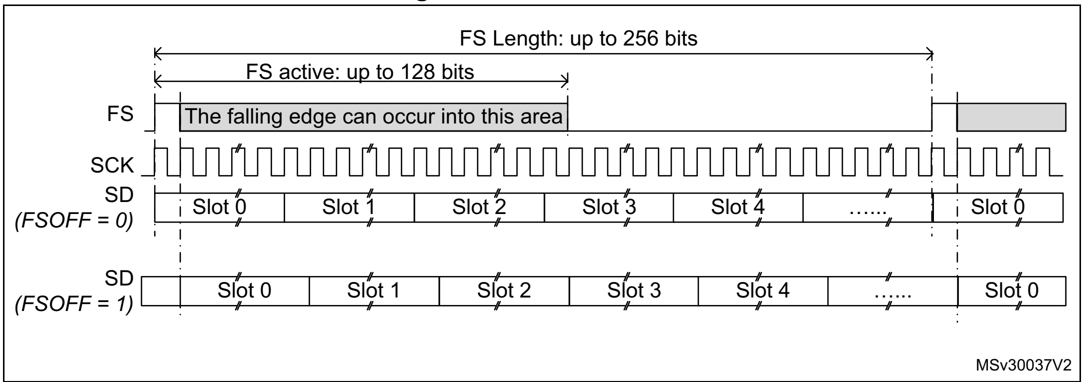
53.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

53.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 607* illustrates this flexibility.

Figure 607. Audio frame



In AC’97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC’97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z depending on the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 53.4.8: SAI clock generator*"

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to *Section 53.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 53.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 53.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

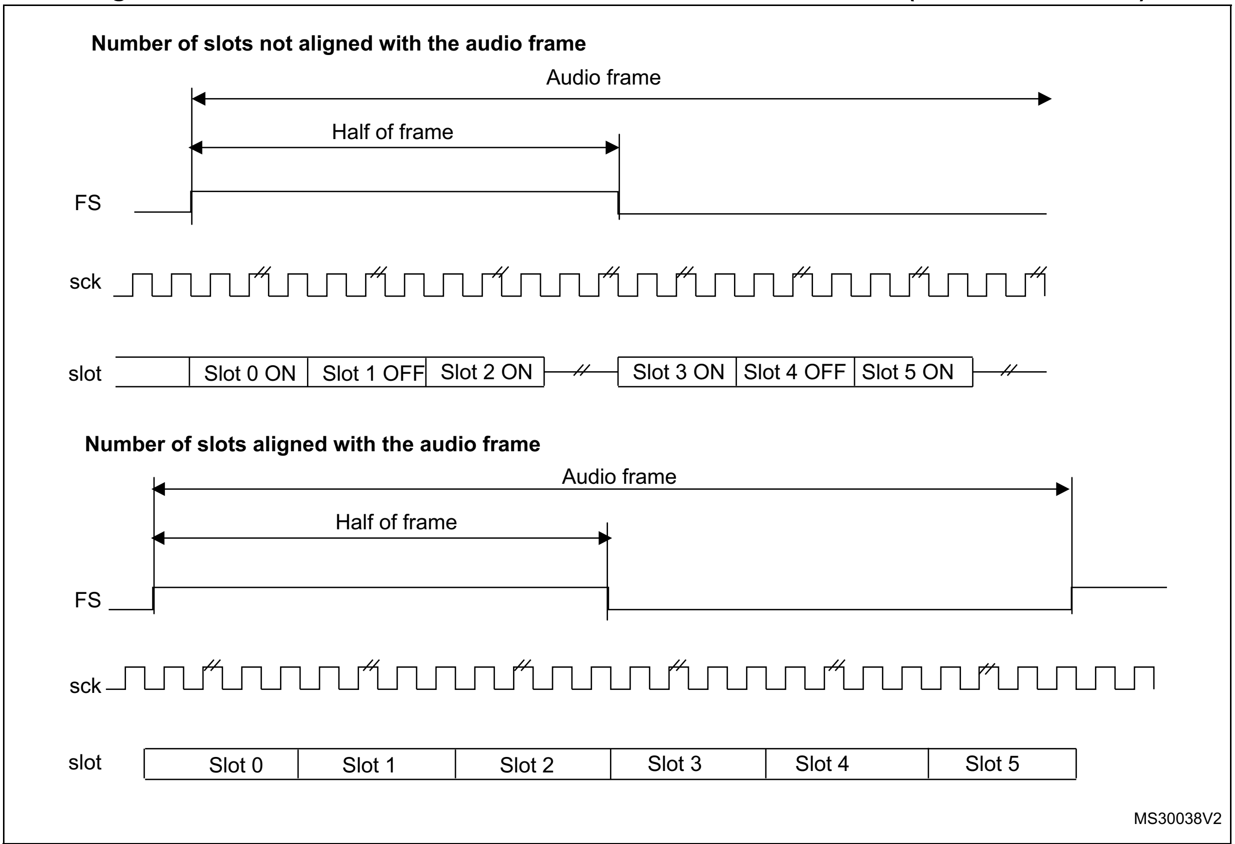
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 608. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

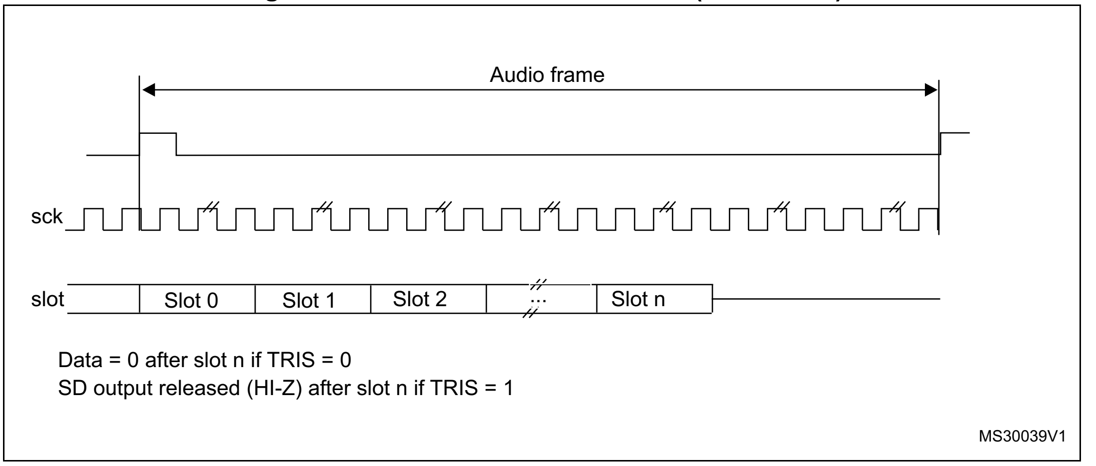


1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 609. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

53.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

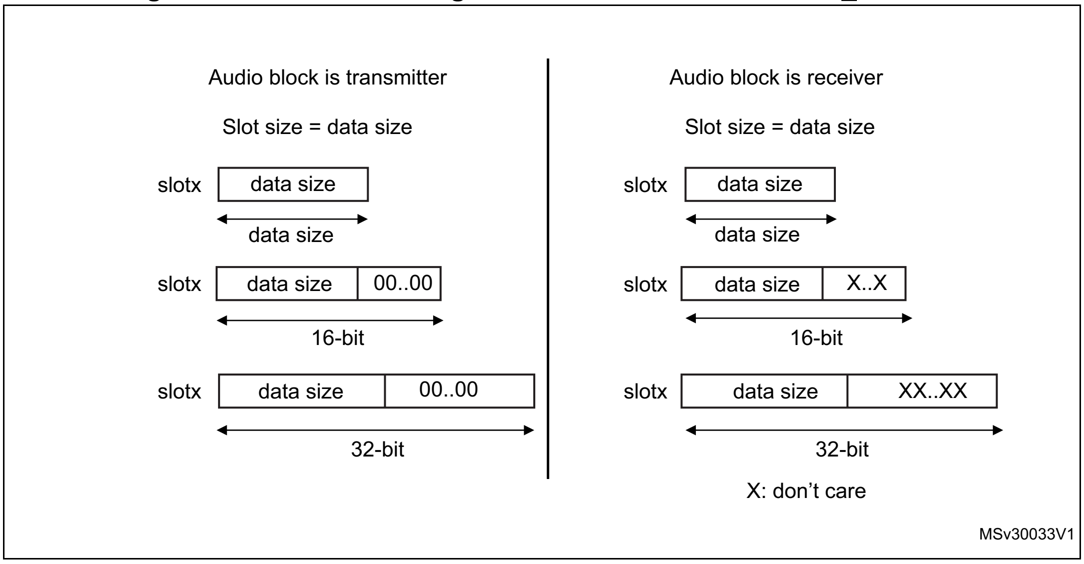
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

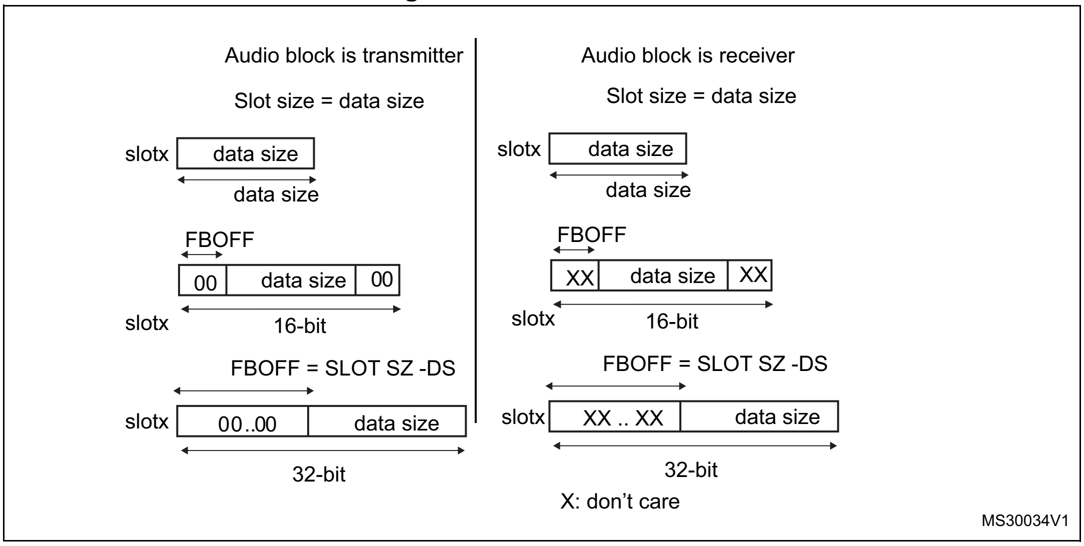
The slot size is also configurable as shown in *Figure 610.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 610. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 611. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤(SLOTSZ - DS)
- DS ≤SLOTSZ
- NBSLOT x SLOTSZ ≤FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 53.4.11: AC'97 link controller*.

53.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 385 shows MCLK_x activation conditions.

Table 385. MCLK_x activation conditions

<table>
<tr><th>MCLKEN</th><th>NODIV</th><th>SAIEN for block x</th><th>MCLK_x</th></tr>
<tr><td>0</td><td rowspan="2">X</td><td rowspan="2">0</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
<tr><td>0</td><td rowspan="2">1</td><td rowspan="3">1</td><td>Disabled</td></tr>
<tr><td>1</td><td>Enabled</td></tr>
<tr><td>X</td><td>0</td><td>Enabled</td></tr>
</table>

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

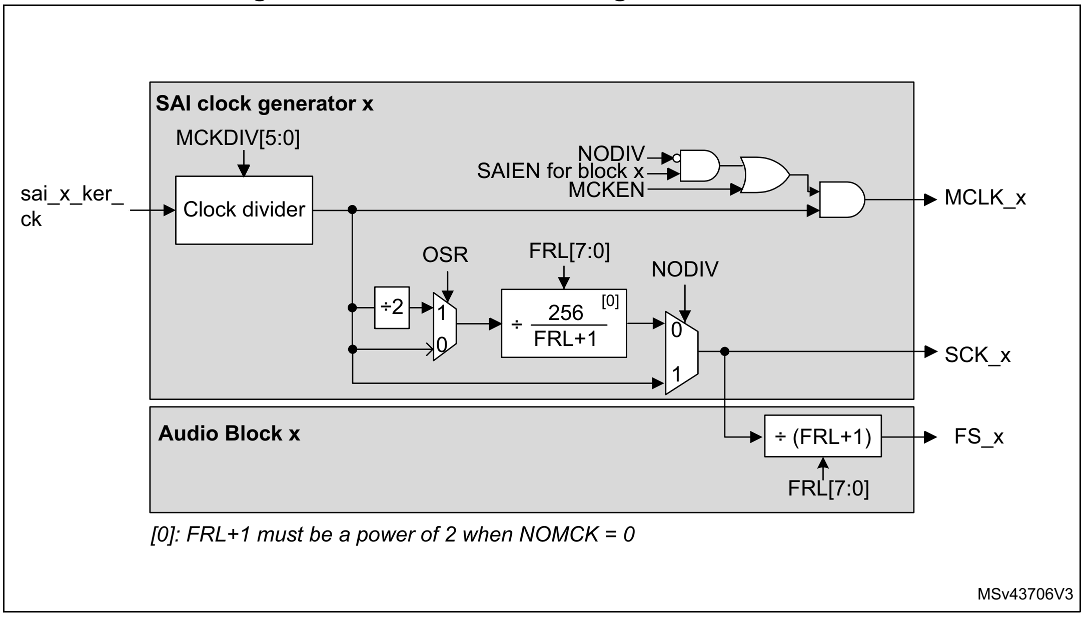
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 612 illustrates the architecture of the audio block clock generator.

Figure 612. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

- F_{MCLK_x} = 256 x F_{FS_x} if OSR = 0
- F_{MCLK_x} = 512 x F_{FS_x} if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^{8}.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK\_x} = F_{MCLK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS\_x} = \frac{F_{sai\_x\_ker\_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 386 gives programming examples for 48, 96, and 192 kHz.

Table 386. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL [(1)]	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio Sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

53.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

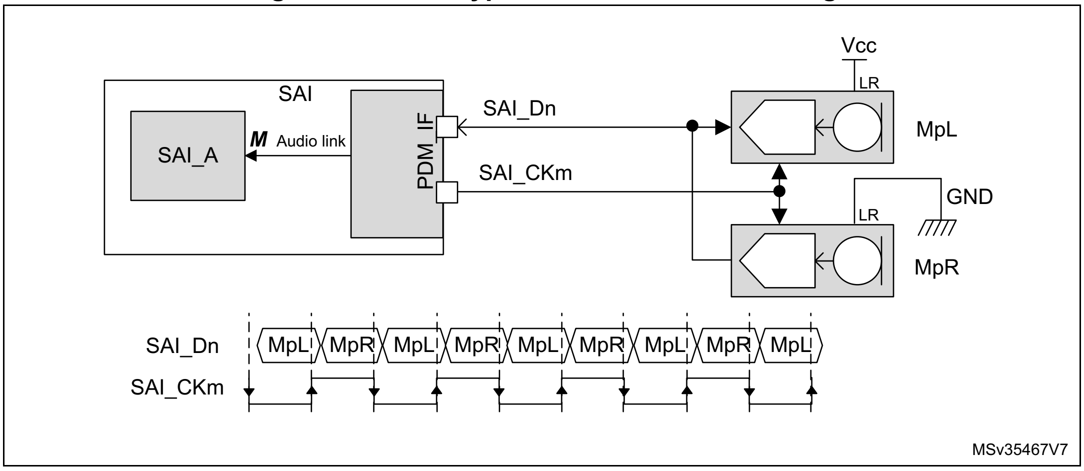
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

53.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to *Section 53.3: SAI implementation*).

Figure 613 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 613. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

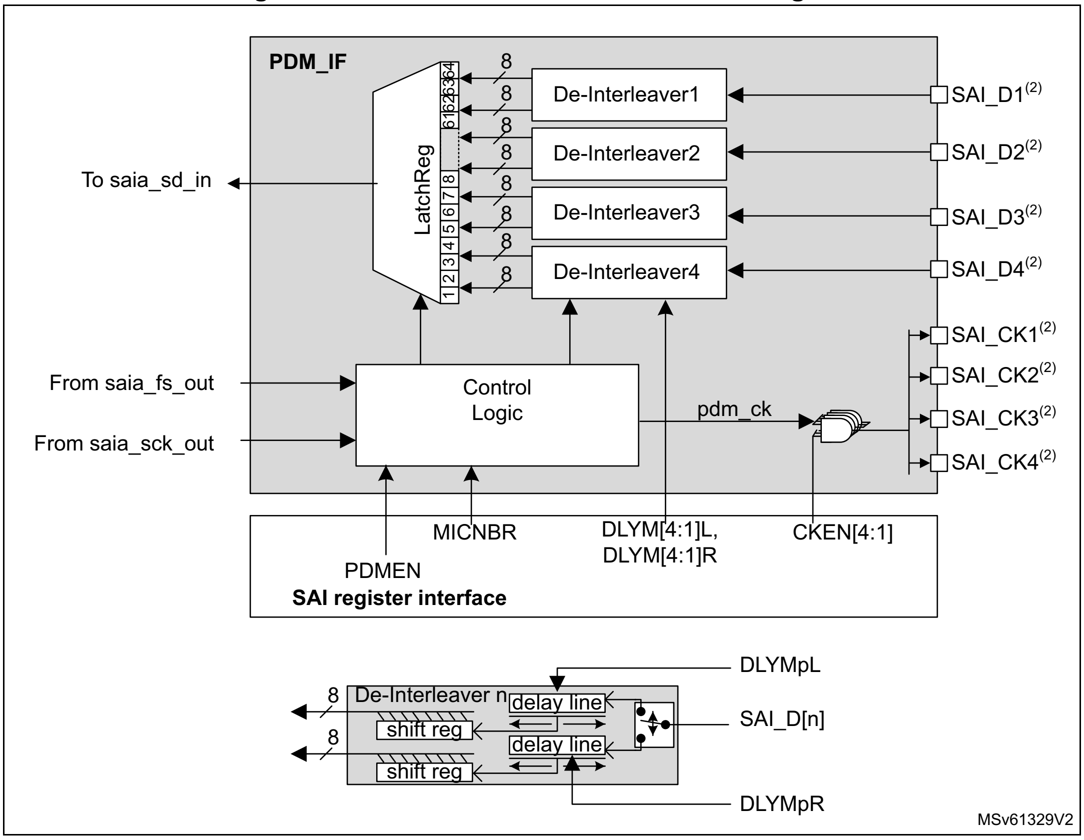
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A through the data line of the serial audio interface.

Figure 614 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 614. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

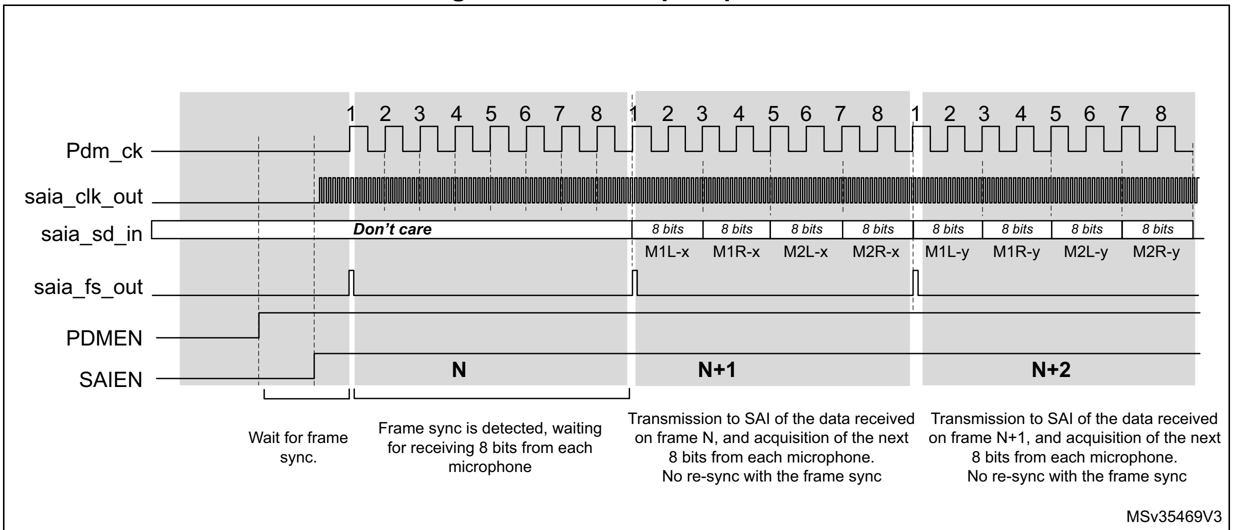
1. Configure SAI_A in TDM master mode (see *Table 387*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 615 shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the serial audio interface.

Figure 615. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

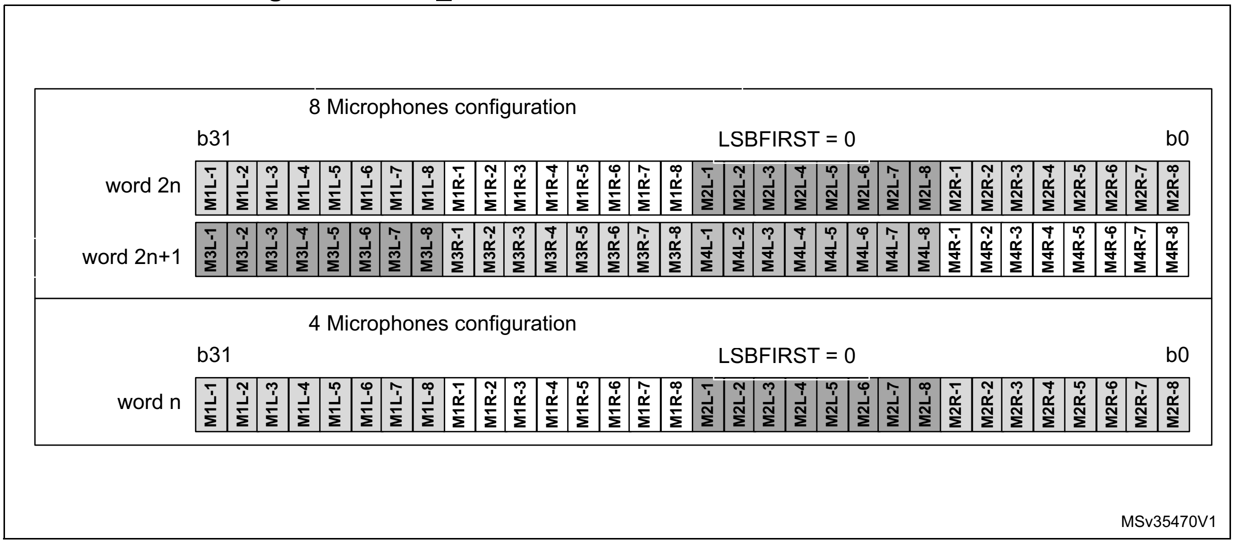
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 616*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 616. SAI_ADR format in TDM mode, 32-bit slot width

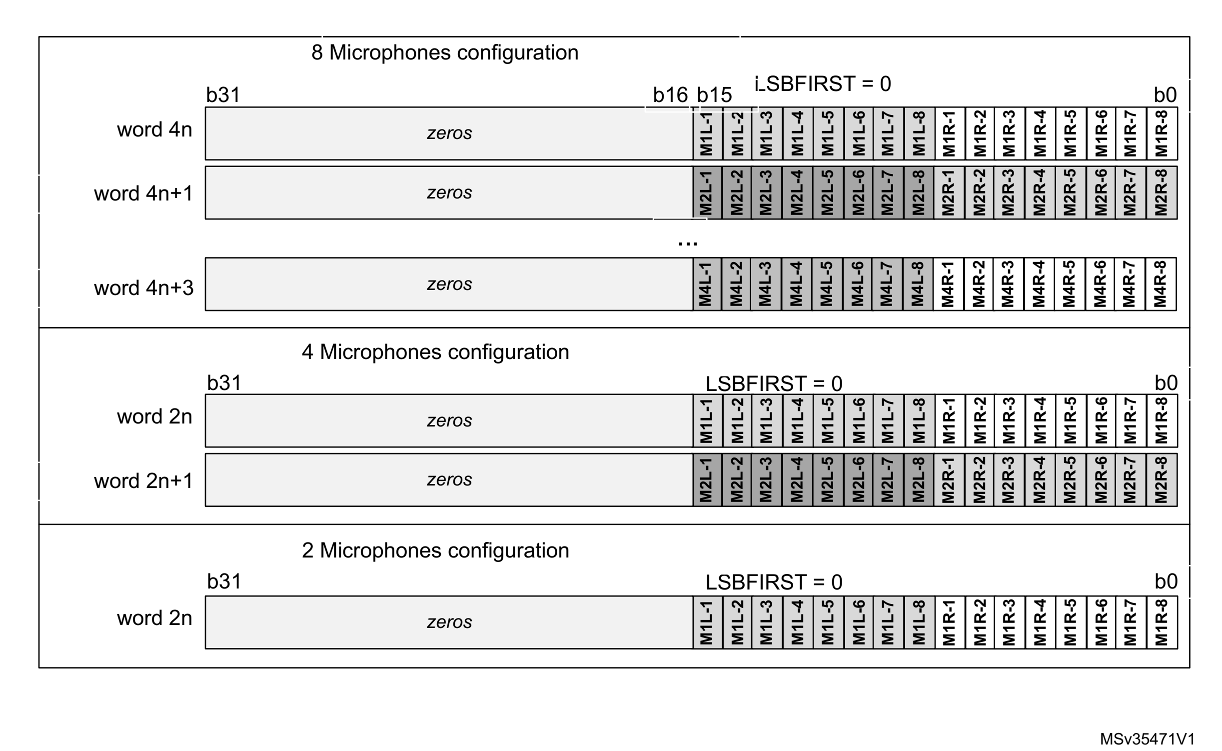


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 617*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 617. SAI_ADR format in TDM mode, 16-bit slot width

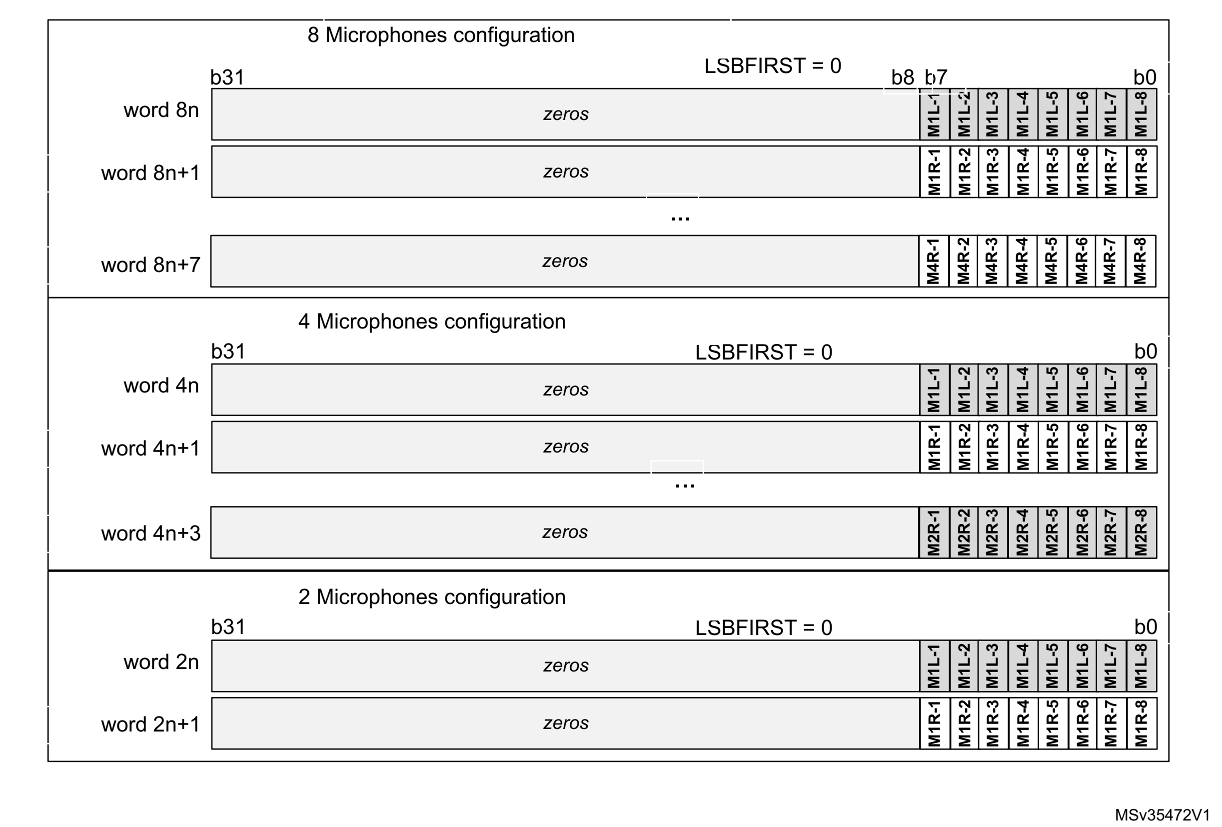


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 618*.

  For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 618. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 387* to ensure a good connection with the PDM interface.

Table 387. SAI_A configuration for TDM mode

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver.
PRTCFG	0b00	Free protocol for TDM.
DS	X	To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See *Table 388*.
LSBFIRST	X	This parameter can be used according to the desired data format.
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode.
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 388*.
FSALL	0	Pulse width is one bit clock cycle.

Table 387. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
FSPOL	1	FS is active high.
FSOFF	0	FS is asserted on the first bit of slot 0.
FBOFF	0	No offset on slot.
SLOTSZ	0	Slot size = data size.
NBSLOT	X	To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See *Table 388*.
SLOTEN	X	To be adjusted according to NBSLOT.
NODIV	1	No need to generate a master clock MCLK.
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 388*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 387*, and follow the sequence below:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK\_A} = F_{PDM\_CK} \times (MICNBR + 1) \times 2$$

MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see *Section 53.6.18)*

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 388. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 387: SAI_A configuration for TDM mode* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV must be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device.*
3. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

53.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

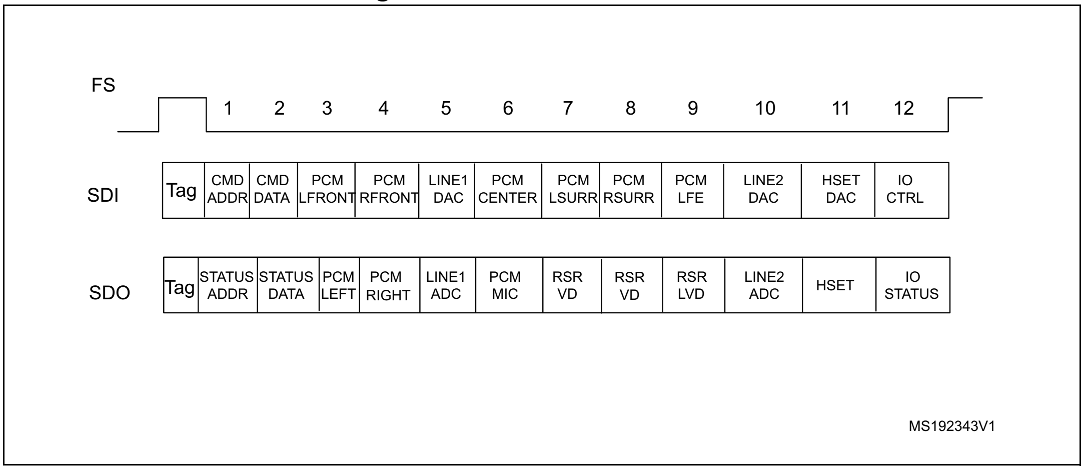
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 619 shows an AC’97 audio frame structure.

Figure 619. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

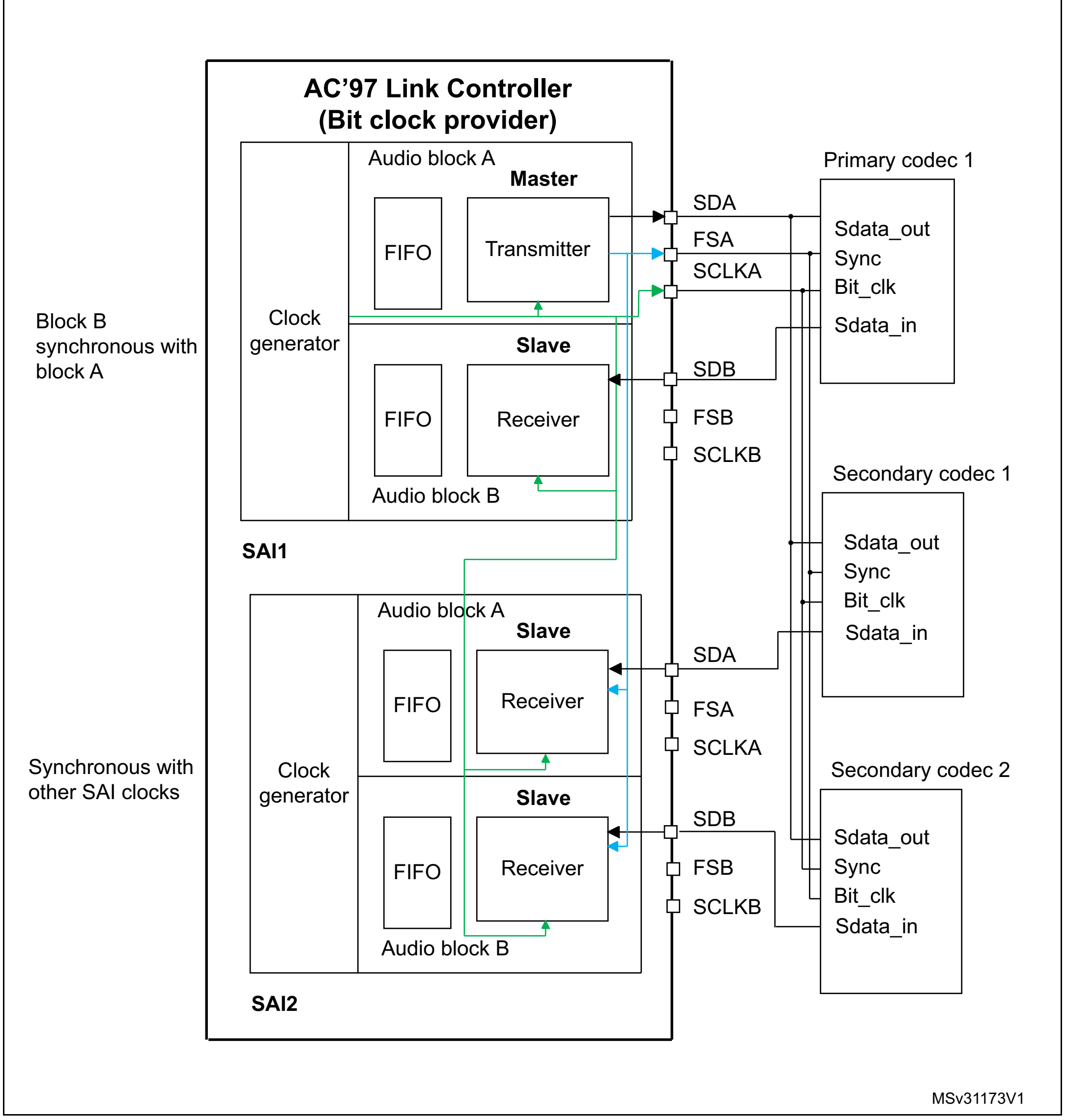
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 620*.

In SAI1, the audio block A must be declared as asynchronous master transmitter, whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.

Figure 620. Example of typical AC'97 configuration on devices featuring at least two embedded SAIs (three external AC'97 decoders)



In receiver mode, the SAI acting as an AC'97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC'97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 53.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

53.4.12 SPDIF output

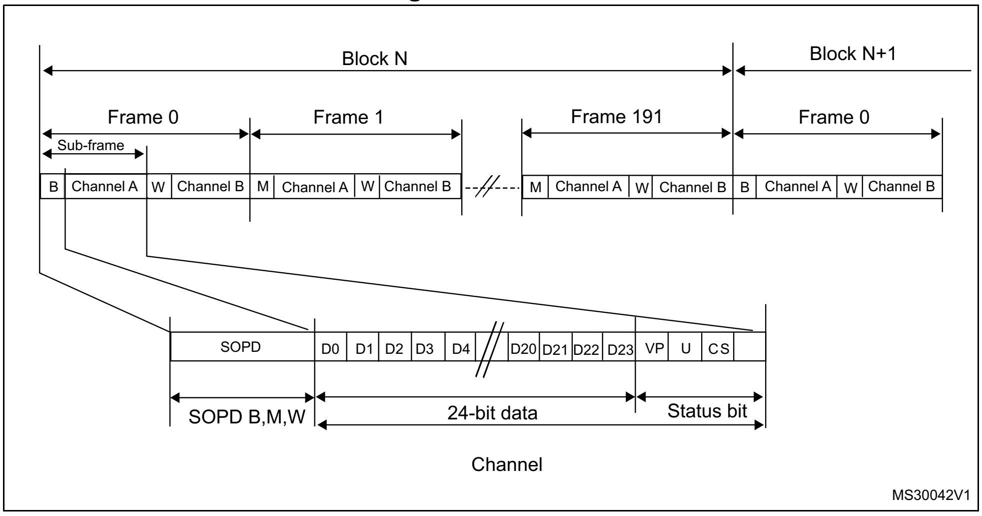
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 621*.

Figure 621. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 389*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 389. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

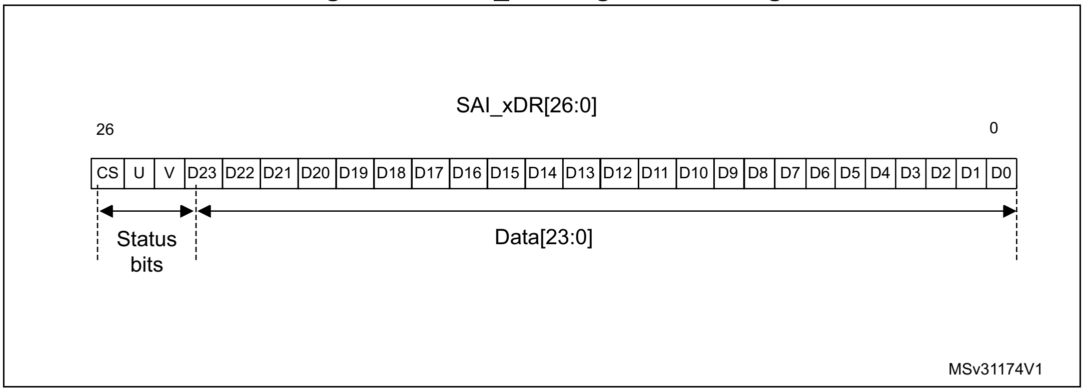
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 622. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 390*.

Table 390. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 391. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_x}) is given by the formula:

$$F_S = \frac{F_{SCK\_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK\_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

53.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 623*. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

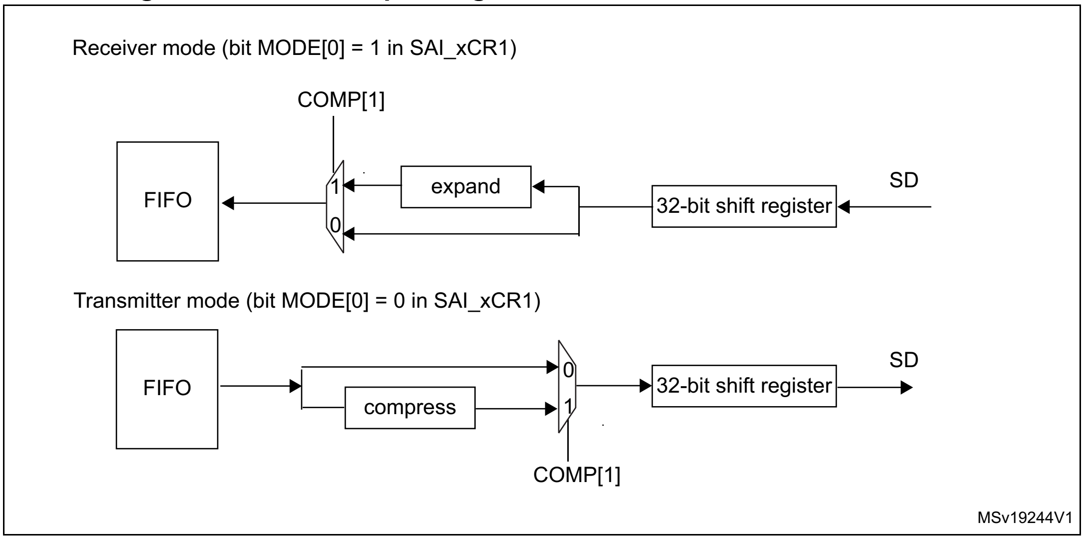
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1’s complement or 2’s complement representation, depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 623. Data companding hardware in an audio block in the SAI



1. Not applicable when AC’97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

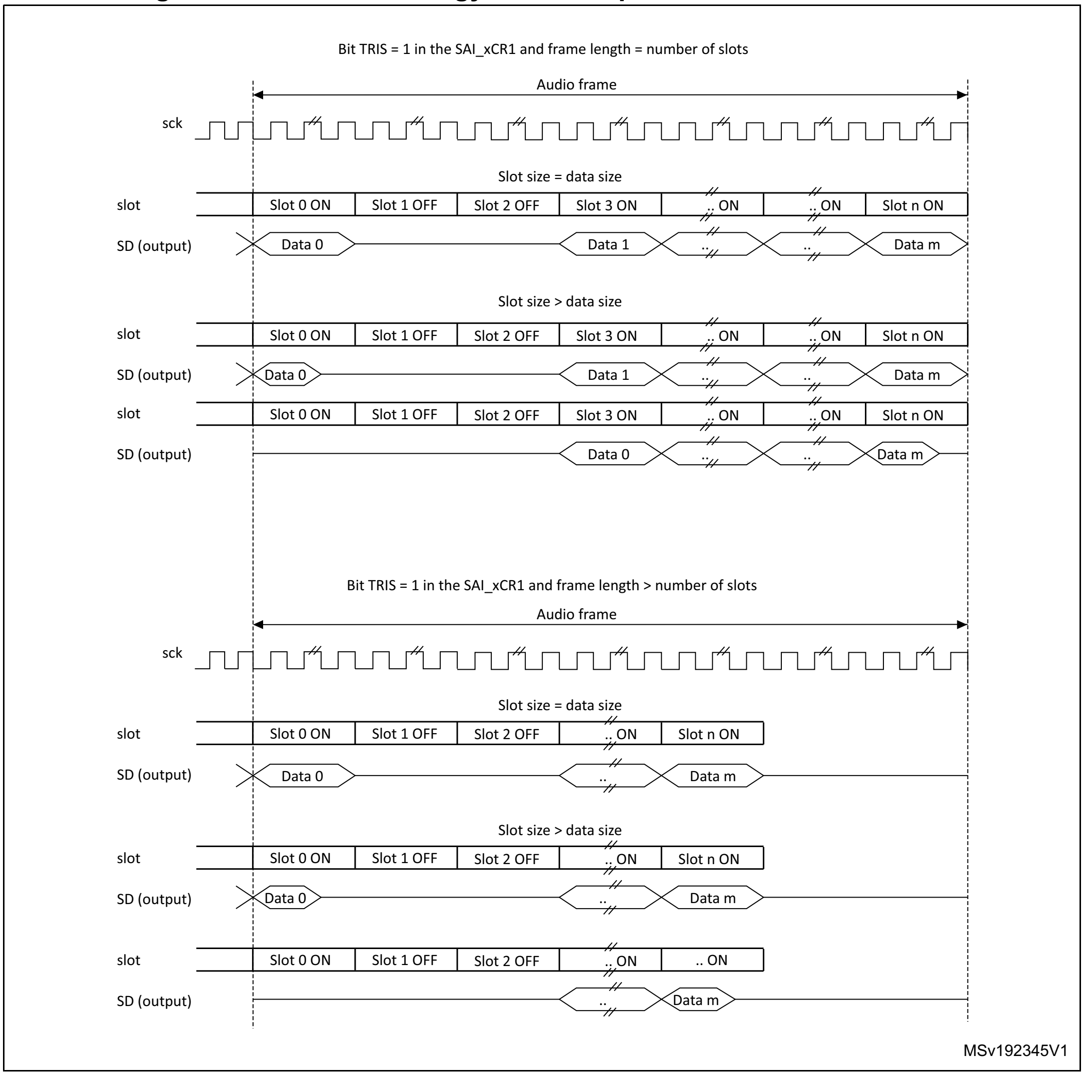
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

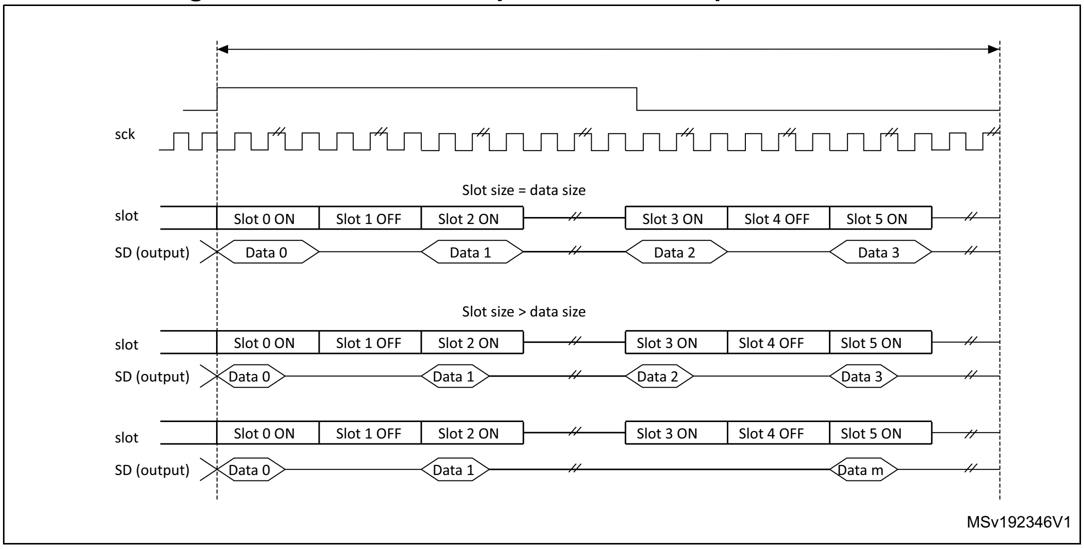
Figure 624 illustrates these behaviors.

Figure 624. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 625* (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 625. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in *Figure 624* and *Figure 625* are replaced by a drive with a value of 0.

53.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC'97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

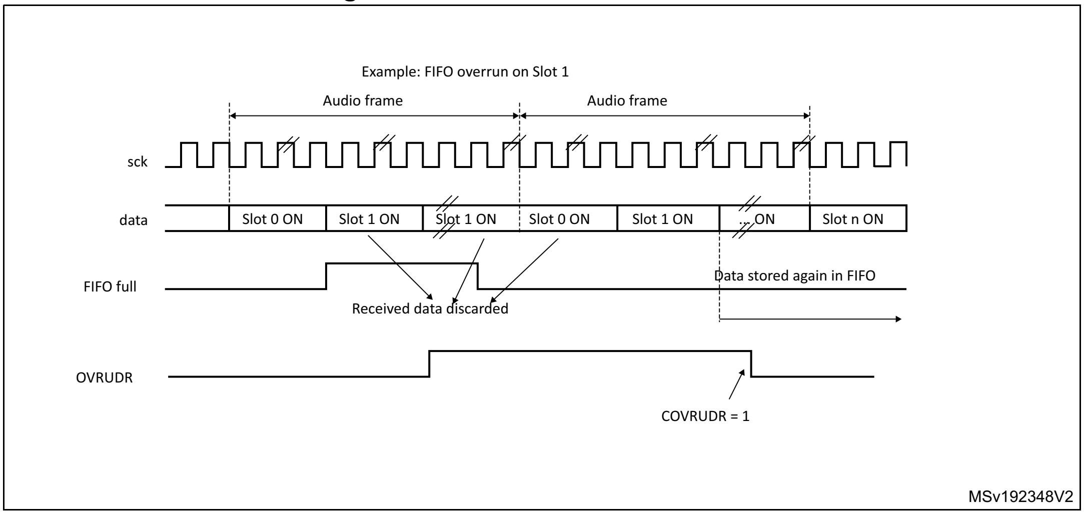
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 626*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 626. Overrun detection error



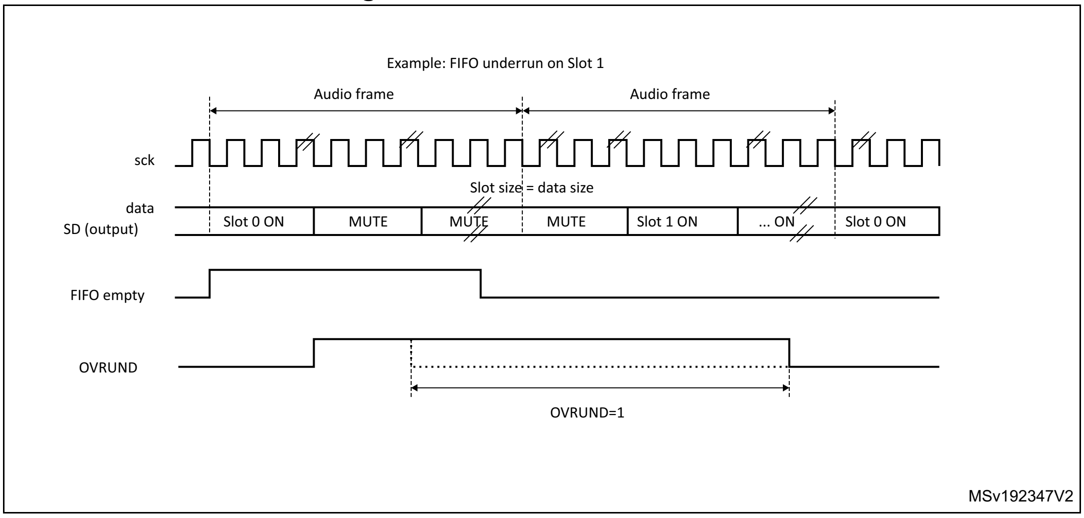
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 627*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 627. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

53.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

53.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 53.4.9: Internal FIFOs*). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

53.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 392*.

Table 392. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 53.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

53.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

53.6.1 SAI global configuration register (SAI_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

00: No synchronization output signals. SYNCOUT[1:0] must be configured as “No synchronization output signals” when audio block is configured as SPDIF

01: Block A used for further synchronization for others SAI

10: Block B used for further synchronization for others SAI

11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to *Table 384: External synchronization selection* for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

53.6.2 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable

0: The master clock is not generated

1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock

This bit is meaningful only when NODIV bit is set to 0.

0: Master clock frequency = F_{FS} x 256

1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider

These bits are set and cleared by software.

000000: Divides by 1 the kernel clock input (sai_x_ker_ck).

Otherwise, The master clock frequency is calculated according to the formula given in *Section 53.4.8: SAI clock generator*.

These bits have no meaning when the audio block is slave.

They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider

This bit is set and cleared by software.

0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512

1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable

This bit is set and cleared by software.

0: DMA disabled

1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable

This bit is set by software.

To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.

This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.

0: SAI audio block disabled

1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive

This bit is set and cleared by software.

0: Audio block output driven when SAIEN is set

1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

53.6.3 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

53.6.4 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.

This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.

This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

53.6.5 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

53.6.6 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET IE	AFSDETI E	CNRDY IE	FREQ IE	WCKCFG IE	MUTEDET IE	OVRUDR IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.

Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

53.6.7 SAI status register (SAI_ASR)

Address offset: 0x18

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
- If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
- If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 53.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

53.6.8 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.

This bit is write only.

Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.

This bit is not used in AC’97 or SPDIF mode

Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.

This bit is write only.

Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.

It is not used in AC’97 or SPDIF mode.

Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.

This bit is write only.

Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.

This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.

Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

53.6.9 SAI data register (SAI_ADR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

53.6.10 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 53.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.

Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.

Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC’97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

53.6.11 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.
It defines the type of complement to be used for companding mode
0: 1's complement representation.
1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.
The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.
Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.
If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.
if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.
Refer to *Section : Mute mode* for more details.
0: Bit value 0 is sent during the mute mode.
1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.
Refer to *Section : Mute mode* for more details.
0: No mute mode.
1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.
Refer to *Section : Output data line management on an inactive slot* for more details.
0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.
0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

53.6.12 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

53.6.13 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size

This bits is set and cleared by software.

The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.

Refer to *Output data line management on an inactive slot* for information on how to drive SD line.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).

01: 16-bit

10: 32-bit

11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset

These bits are set and cleared by software.

The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.

These bits must be set when the audio block is disabled.

They are ignored in AC’97 or SPDIF mode.

53.6.14 SAI interrupt mask register (SAI_BIM)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.

This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.

This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.

This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.

Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.

It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.

Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.

This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.

This bit is set and cleared by software.

0: Interrupt is disabled

1: Interrupt is enabled

When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

53.6.15 SAI status register (SAI_BSR)

Address offset: 0x38

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).

000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.
0: External AC’97 Codec is ready
1: External AC’97 Codec is not ready
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:

– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.

This bit is read only.

0: Clock configuration is correct

1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 53.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)

This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.

It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.

This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.

This bit is read only.

0: No MUTE detection on the SD input line

1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame

This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).

It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.

This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.

This bit is read only.

0: No overrun/underrun error.

1: Overrun/underrun error detection.

The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.

It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.

This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

53.6.16 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

53.6.17 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data

A write to this register loads the FIFO provided the FIFO is not full.

A read from this register empties the FIFO if the FIFO is not empty.

53.6.18 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
				rw	rw	rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **CKEN4**: Clock enable of bitstream clock number 4

This bit is set and cleared by software.

0: SAI_CK4 clock disabled

1: SAI_CK4 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK4 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bit 10 **CKEN3**: Clock enable of bitstream clock number 3

This bit is set and cleared by software.

0: SAI_CK3 clock disabled

1: SAI_CK3 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK3 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI_CK2 clock disabled

1: SAI_CK2 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK2 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI_CK1 clock disabled

1: SAI_CK1 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK1 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.**

The complete set of data lines might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

53.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 of T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

53.6.20 SAI hardware configuration register (SAI_HWCFGR)

Address offset: 0x03F0

Reset value: 0x0000 0108

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTION_REGOUT[7:4]			
												r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OPTION_REGOUT[3:0]				SPDIF_PDM[3:0]				FIFO_SIZE[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:12 **OPTION_REGOUT[7:0]:** Support of SAI_IOR register
0: SAI_IOR register is not implemented.
1: SAI_IOR register is implemented, with only 1 control bit at position 0.
2: SAI_IOR register is implemented, with 2 control bits at positions 0 and 1.
...
16: SAI_IOR register is implemented, with 16 control bits at positions 0 to 15.

Bits 11:8 **SPDIF_PDM[3:0]:** Support of SPDIF-OUT and PDM interfaces
0: SPDIF-OUT and PDM interfaces are not available.
1: SPDIF-OUT and PDM interfaces are available.

Bits 7:0 **FIFO_SIZE[7:0]:** FIFO size for SAIA and SAIB
4: the FIFO depth is 4 words.
8: the FIFO depth is 8 words.
12: the FIFO depth is 12 words.
16: the FIFO depth is 16 words.
20: the FIFO depth is 20 words.
24: the FIFO depth is 24 words.
28: the FIFO depth is 28 words.
32: the FIFO depth is 32 words.
others: Invalid values

53.6.21 SAI version register (SAI_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** SAI major revision

Bits 3:0 **MINREV[3:0]:** SAI minor revision

53.6.22 SAI identification register (SAI_IPIDR)

Address offset: 0x03F8

Reset value: 0x0013 0031

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: SAI identifier

53.6.23 SAI size identification register (SAI_SIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits returns the size of the memory region allocated to SAI registers.
The size of this memory region is of 1 Kbytes.

53.6.24 SAI register map

Table 393. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
	Reset value																											0	0			0	0
0x04 or 0x24	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0

Table 393. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08 or 0x28	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]						MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C or 0x2C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0
0x14 or 0x34	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x18 or 0x38	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	1	0	0	0
0x1C or 0x3C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x20 or 0x40	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN4	CKEN3	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																					0	0	0	0			0	0				0
0x48	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x004C to 0x03EC	Reserved																																
0x03F0	SAI_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTION_REGOUT[7:0]								SPDIF_ PDM[3:0]				FIFO_SIZE[7:0]							
	Reset value													0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
0x03F4	SAI_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value	0x0000 0021																															

Table 393. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03F8	SAI_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
0x03FC	SAI_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5: Memory organization* for the register boundary addresses.

54 SPDIF receiver interface (SPDIFRX)

54.1 SPDIFRX interface introduction

The SPDIFRX interface handles S/PDIF audio protocol.

54.2 SPDIFRX main features

- Up to 4 inputs available
- Automatic symbol rate detection
- Maximum symbol rate: 12.288 MHz
- Stereo stream from 8 to 192 kHz[a] supported
- Supports audio IEC-60958 and IEC-61937, consumer applications
- Parity bit management
- Communication using DMA for audio samples
- Communication using DMA for control and user channel information
- Interrupt capabilities

54.3 SPDIFRX functional description

The SPDIFRX peripheral, is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multi-channel surround sound, such as those defined by Dolby or DTS.

The receiver provides all the necessary features to detect the symbol rate, and decode the incoming data. It is possible to use a dedicated path for the user and channel information in order to ease the interface handling. *Figure 628* shows a simplified block diagram.

The SPDIFRX_DC block is responsible of the decoding of the S/PDIF stream received from SPDIFRX_IN[3:0] inputs. This block re-sample the incoming signal, decode the manchester stream, recognize frames, sub-frames and blocks elements. It delivers to the REG_IF part, decoded data, and associated status flags.

This peripheral can be fully controlled via the APB bus, and can handle two DMA channels:

- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user information

Interrupt services are also available either as an alternative function to the DMA, or for signaling error or key status of the peripheral.

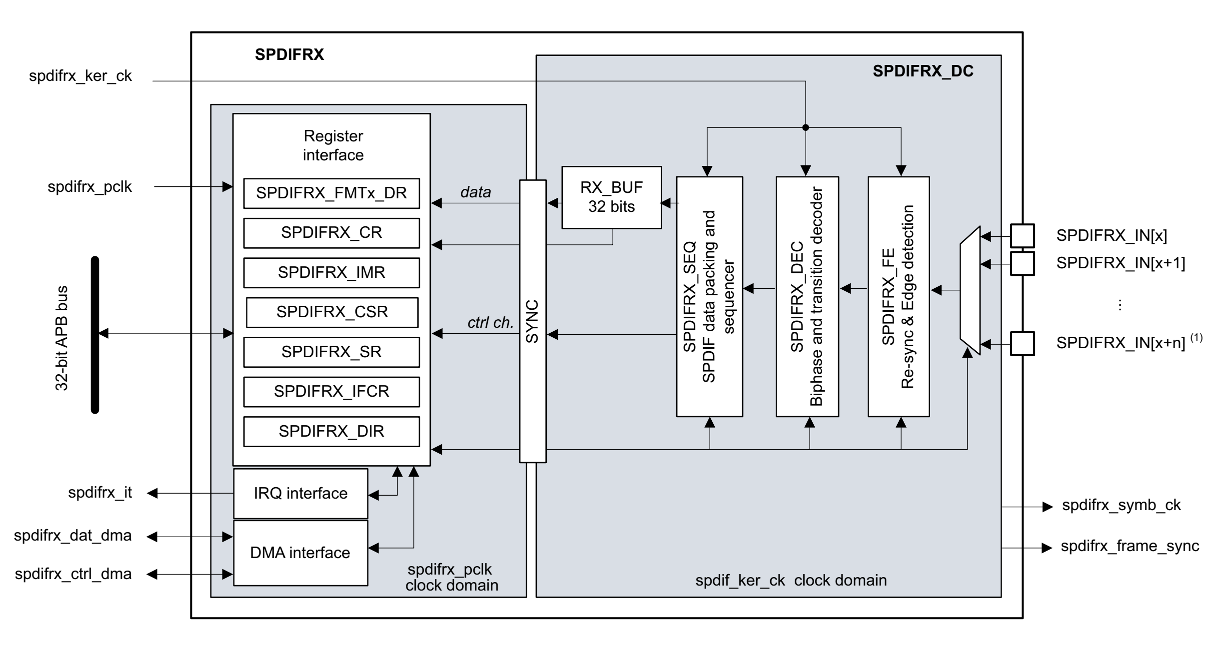
The SPDIFRX also offers a signal named spdifrx_frame_sync, which toggles every time that a sub-frame's preamble is detected. So the duty cycle is 50%, and the frequency equal to the frame rate.

This signal can be connected to timer events, in order to compute frequency drift.

a. Check the RCC capabilities in order to verify which sampling rates can be supported.

In addition the SPDIFRX also provides a signal named **spdifrx_symb_ck** toggling at the symbol rate.

Figure 628. SPDIFRX block diagram



1. 'n' is fixed to 3, and 'x' is set to 0.

54.3.1 SPDIFRX pins and internal signals

Table 394 lists the SPDIFRX internal input/output signals, *Table 395* the SPDIFRX pins (alternate functions).

Table 394. SPDIFRX internal input/output signals

Signal name	Signal type	Description
spdifrx_ker_ck	Digital input	SPDIFRX kernel clock
spdifrx_pclk	Digital input	SPDIFRX register interface clock
spdifrx_it	Digital output	SPDIFRX global interrupt
spdifrx_dat_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for data transfer
spdifrx_ctrl_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for channel status and user information transfer
spdifrx_frame_sync	Digital output	SPDIFRX frame rate synchronization signal
spdifrx_symb_ck	Digital output	SPDIFRX channel symbol clock

Table 395. SPDIFRX pins

Pin name	Pin type	Description
SPDIFRX_IN0	Digital input	Input 0 for S/PDIF signal
SPDIFRX_IN1	Digital input	Input 1 for S/PDIF signal

Table 395. SPDIFRX pins (continued)

Pin name	Pin type	Description
SPDIFRX_IN2	Digital input	Input 2 for S/PDIF signal
SPDIFRX_IN3	Digital input	Input 3 for S/PDIF signal

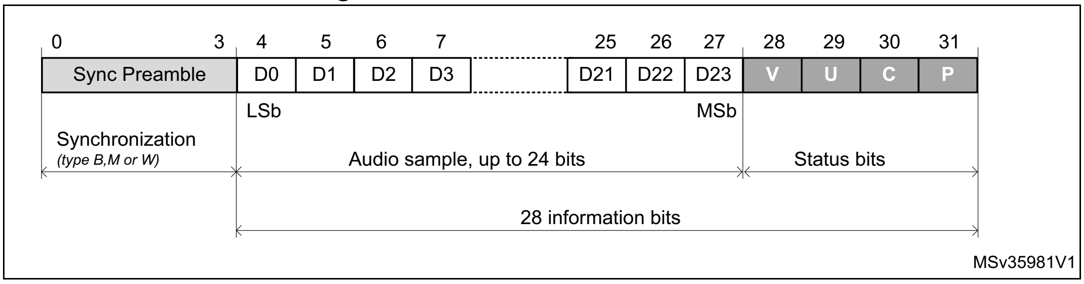
54.3.2 S/PDIF protocol (IEC-60958)

S/PDIF block

A S/PDIF frame is composed of two sub-frames (see *Figure 629*). Each sub-frame contains 32 bits (or time slots):

- Bits 0 to 3 carry one of the synchronization preambles
- Bits 4 to 27 carry the audio sample word in linear 2's complement representation. The most significant bit (MSB) is carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8.
- Bit 28 (validity bit "V") indicates if the data is valid (for converting it to analog for example)
- Bit 29 (user data bit "U") carries the user data information like the number of tracks of a Compact Disk.
- Bit 30 (channel status bit "C") carries the channel status information like sample rate and protection against copy.
- Bit 31 (parity bit "P") carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity).

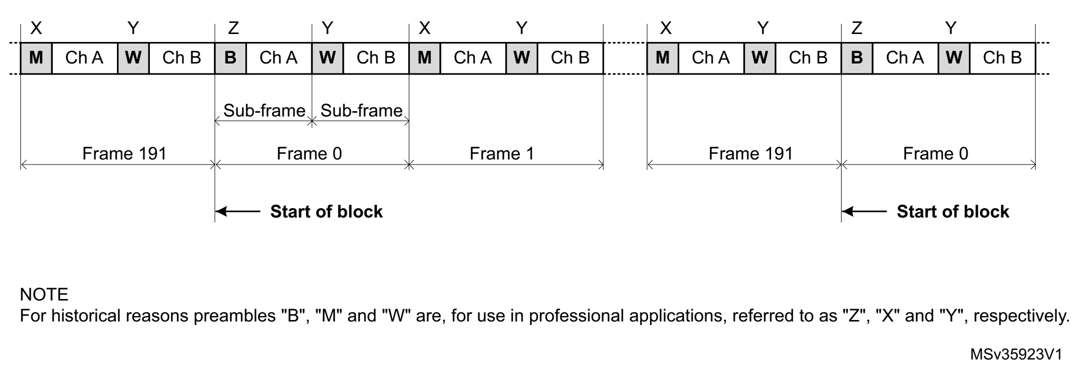
Figure 629. S/PDIF sub-frame format



For linear coded audio applications, the first sub-frame (left or "A" channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble "M". However, the preamble changes to preamble "B" once every 192 frames to identify the start of the block structure used to organize the channel status and user information. The second sub-frame (right or "B" channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble "W".

A S/PDIF block contains 192 pairs of sub-frames of 32 bits.

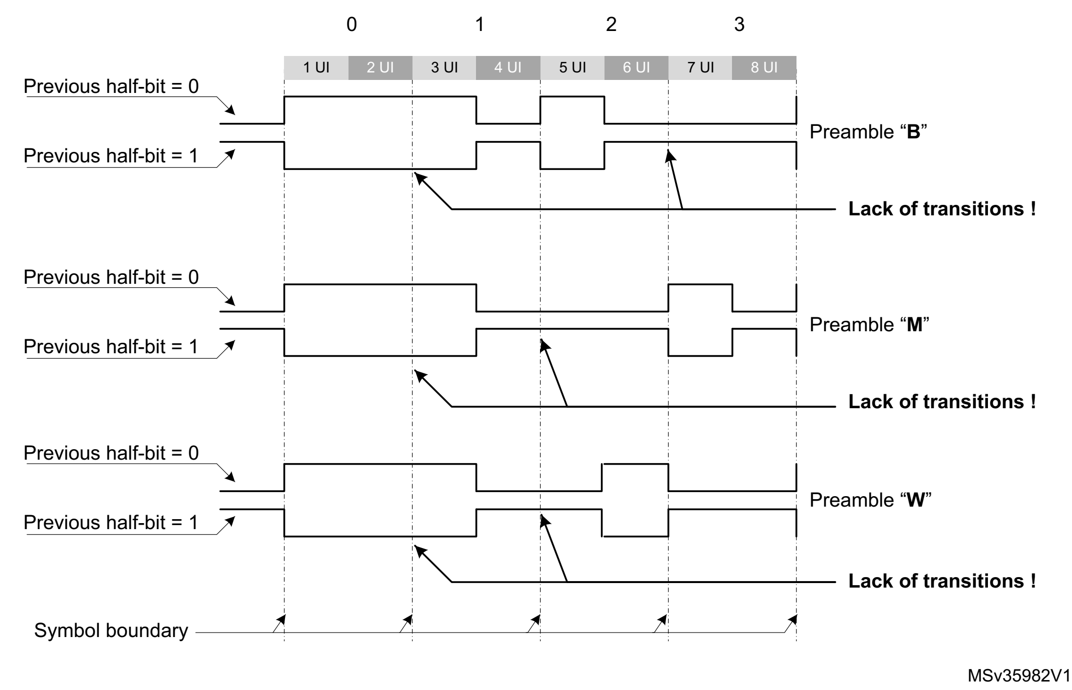
Figure 630. S/PDIF block format



Synchronization preambles

The preambles patterns are inverted or not according to the previous half-bit value. This previous half-bit value is the level of the line before enabling a transfer for the first “B” preamble of the first frame. For the others preambles, this previous half-bit value is the second half-bit of the parity bit of the previous sub-frame. The preambles patterns B, M and W are described in the *Figure 631*.

Figure 631. S/PDIF Preambles



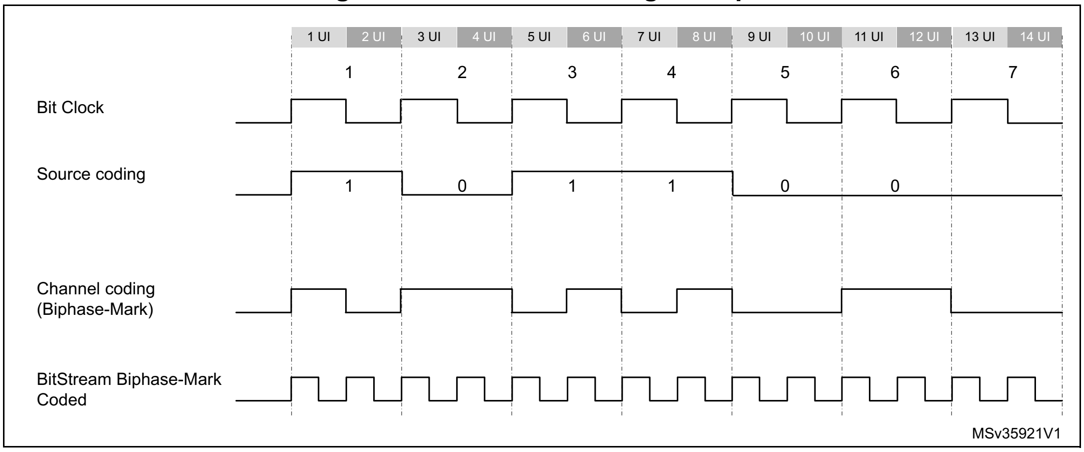
Coding of information bits

In order to minimize the DC component value on the transmission line, and to facilitate clock recovery from the data stream, bits 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is logical 0. However, it is different if the bit is logical 1. These states are named “UI” (unit interval) in the IEC-60958 specification.

The 24 data bits are transferred LSB first.

Figure 632. Channel coding example



54.3.3 SPDIFRX decoder (SPDIFRX_DC)

Main principle

The technique used by the SPDIFRX in order to decode the S/PDIF stream is based on the measurement of the time interval between two consecutive edges. Three kinds of time intervals may be found into an S/PDIF stream:

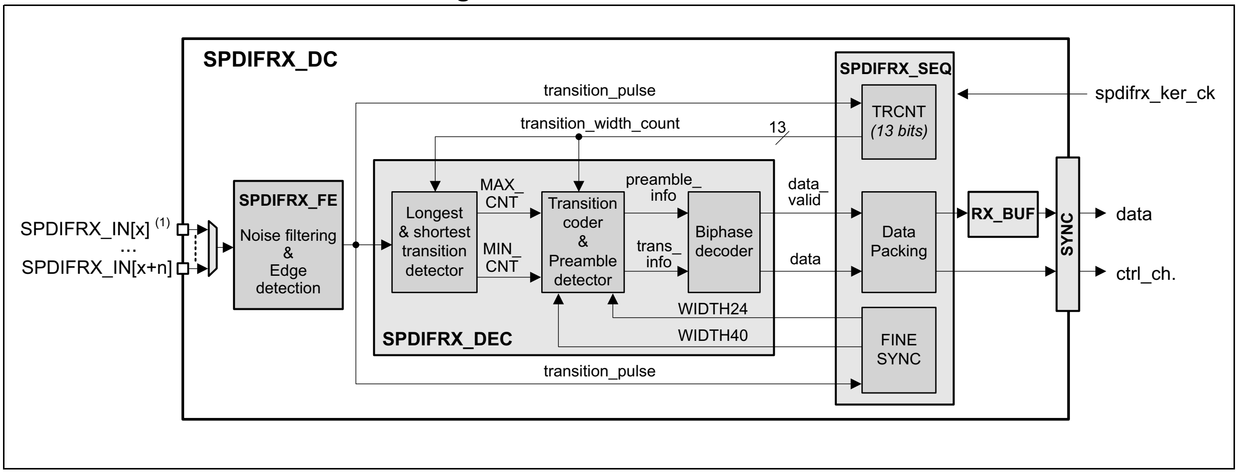
- The long time interval, having a duration of 3 x UI, noted TL. It appears only during preambles.
- The medium time interval, having a duration of 2 x UI, noted TM. It appears both in some preambles or into the information field.
- The short time interval, having a duration of 1 x UI, noted TS. It appears both in some preambles or into the information field.

The SPDIFRX_DC block is responsible of the decoding of the received S/PDIF stream. It takes care of the following functions:

- Resampling and filtering of the incoming signal
- Estimation of the time-intervals
- Estimation of the symbol rate and synchronization
- Decoding of the serial data, and check of integrity
- Detection of the block, and sub-frame preambles
- Continuous tracking of the symbol rate

Figure 633 gives a detailed view of the SPDIFRX decoder.

Figure 633. SPDIFRX decoder



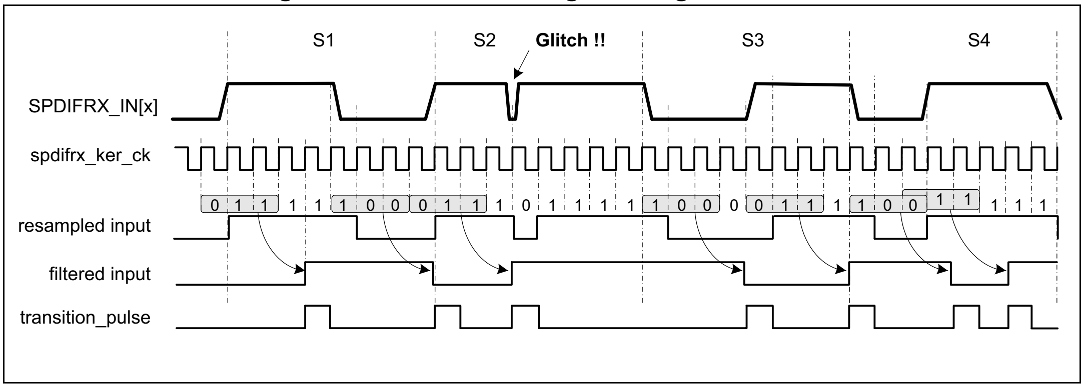
1. 'n' is fixed to 3, and 'x' is set to 0.

Noise filtering and rising/falling edge detection

The S/PDIF signal received on the selected SPDIFRX_IN is re-sampled using the spdifrx_ker_ck clock (acquisition clock). A simple filtering is applied in order cancel spurs. This is performed by the stage detecting the edge transitions. The edge transitions are detected as follow:

- A rising edge is detected when the sequence 0 followed by two 1 is sampled.
- A falling edge is detected when the sequence 1 followed by two 0 is sampled.
- After a rising edge, a falling edge sequence is expected.
- After a falling edge, a rising edge sequence is expected.

Figure 634. Noise filtering and edge detection



Longest and shortest transition detector

The **longest and shortest transition detector** block detects the maximum (MAX_CNT) and minimum (MIN_CNT) duration between two transitions. The TRCNT counter is used to measure the time interval duration. It is clocked by the spdifrx_ker_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again.

The maximum duration is normally found during the preamble period. This maximum duration is sent out as MAX_CNT. The minimum duration is sent out as MIN_CNT.

The search of the longest and shortest transition is stopped when the transition timer expires. The transition timer is like a watchdog timer that generates a trigger after 70 transitions of the incoming signal. Note that counting 70 transitions insures a delay a bit longer than a sub-frame.

Note that when the TRCNT overflows due to a too long time interval between two pulses, the SPDIFRX is stopped and the flag TERR of SPDIFRX_SR register is set to 1.

Transition coder and preamble detector

The **transition coder and preamble detector** block receives the MAX_CNT and MIN_CNT. It also receives the current transition width from the TRCNT counter (see *Figure 633*). This block encodes the current transition width by comparing the current transition width with two different thresholds, names TH_{HI} and TH_{LO}.

- If the current transition width is less than (TH_{LO} - 1), then the data received is half part of data bit '1', and is coded as TS.
- If the current transition width is greater than (TH_{LO} - 1), and less than TH_{HI}, then the data received is data bit '0', and is coded as TM.
- If the current transition width is greater than TH_{HI}, then the data received is the long pulse of preambles, and is coded as TL.
- Else an error code is generated (FERR flag is set).

The thresholds TH_{HI} and TH_{LO} are elaborated using two different methods.

If the peripheral is doing its initial synchronization ('coarse synchronization'), then the thresholds are computed as follow:

- TH_{LO} = MAX_CNT / 2.
- TH_{HI} = MIN_CNT + MAX_CNT / 2.

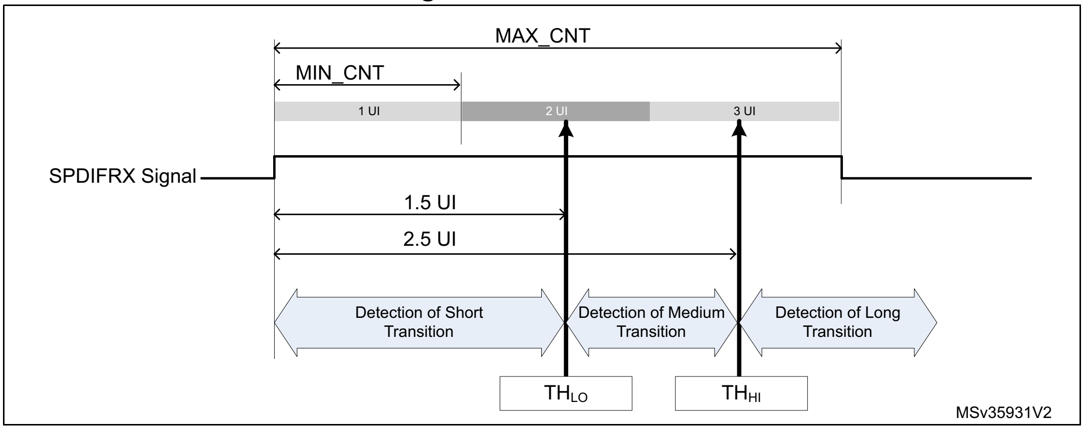
Once the 'coarse synchronization' is completed, then the SPDIFRX uses a more accurate reference in order to elaborate the thresholds. The SPDIFRX measures the length of 24 symbols (WIDTH24) for defining TH_{LO} and the length of 40 symbols (WIDTH40) for TH_{HI}. TH_{HI} and TH_{LO} are computed as follow:

- TH_{LO} = (WIDTH24) / 32
- TH_{HI} = (WIDTH40) / 32

This second synchronization phase is called the 'fine synchronization'. Refer to *Figure 637* for additional information.

As shown in the figure hereafter, TH_{LO} is ideally equal to 1.5 UI, and to TH_{HI} 2.5 UI.

Figure 635. Thresholds



The preamble detector checks four consecutive transitions of a specific sequence to determine if they form the part of preamble. Let us say TRANS0, TRANS1, TRANS2 and TRANS3 represent four consecutive transitions encoded as mentioned above. *Table 396* shows the values of these four transitions to form a preamble. Absence of this pattern indicates that these transitions form part of the data in the sub frame and bi-phase decoder decode them.

Table 396. Transition sequence for preamble

Preamble type	Biphase data pattern	TRANS3	TRANS2	TRANS1	TRANS0
Preamble B	11101000	TL	TS	TS	TL
Preamble M	11100010	TL	TL	TS	TS
Preamble W	11100100	TL	TM	TS	TM

Bi-phase decoder

The Bi-phase decoder decodes the input bi-phase marked data stream using the transition information provided by the **transition coder and preamble detector** block. It first waits for the preamble detection information. After the preamble detection, it decodes the following transition information:

- If the incoming transition information is TM then it is decoded as a ‘0’.
- Two consecutive TS are decoded as a ‘1’.
- Any other transition sequence generates an error signal (FERR set to 1).

After decoding 28 data bits this way, this module looks for the following preamble data. If the new preamble is not what is expected, then this block generates an error signal (FERR set to 1). Refer to *Section 54.3.9: Reception errors*, for additional information on error flags.

Data packing

This block is responsible of the decoding of the IEC-60958 frames and blocks. It also handles the writing into the RX_BUF or into SPDIFRX_CSR register.

54.3.4 SPDIFRX tolerance to clock deviation

The SPDIFRX tolerance to clock deviation depends on the number of sample clock cycles in one bit slot. The fastest spdifrx_ker_ck is, the more robust the reception is. The ratio between spdifrx_ker_ck frequency and the symbol rate must be at least 11.

Two kinds of phenomenon (at least) can degrade the reception quality:

- The cycle-to-cycle jitter which reflects the difference of transition length between two consecutive transitions.
- The long term jitter which reflects a cumulative effect of the cycle-to-cycle jitter. It can be seen as a low-frequency symbol modulation.

54.3.5 SPDIFRX synchronization

The synchronization phase starts when setting SPDIFRXEN to 01 or 11. *Figure 636* shows the synchronization process.

If the bit WFA of SPDIFRX_CR register is set to 1, then the peripheral must first detect activity on the selected SPDIFRX_IN line before starting the synchronization process. The activity detection is performed by detecting four transitions on the selected SPDIFRX_IN. The peripheral remains in this state until transitions are not detected. This function can be particularly helpful because the SPDIFRX switches in COARSE SYNC mode only if activity is present on the selected SPDIFRX_IN input, avoiding synchronization errors. See *Section 54.4: Programming procedures* for additional information.
The user can still set the SPDIFRX into STATE_IDLE by setting SPDIFRXEN to 0. If the WFA is set to 0, the peripheral starts the coarse synchronization without checking activity.

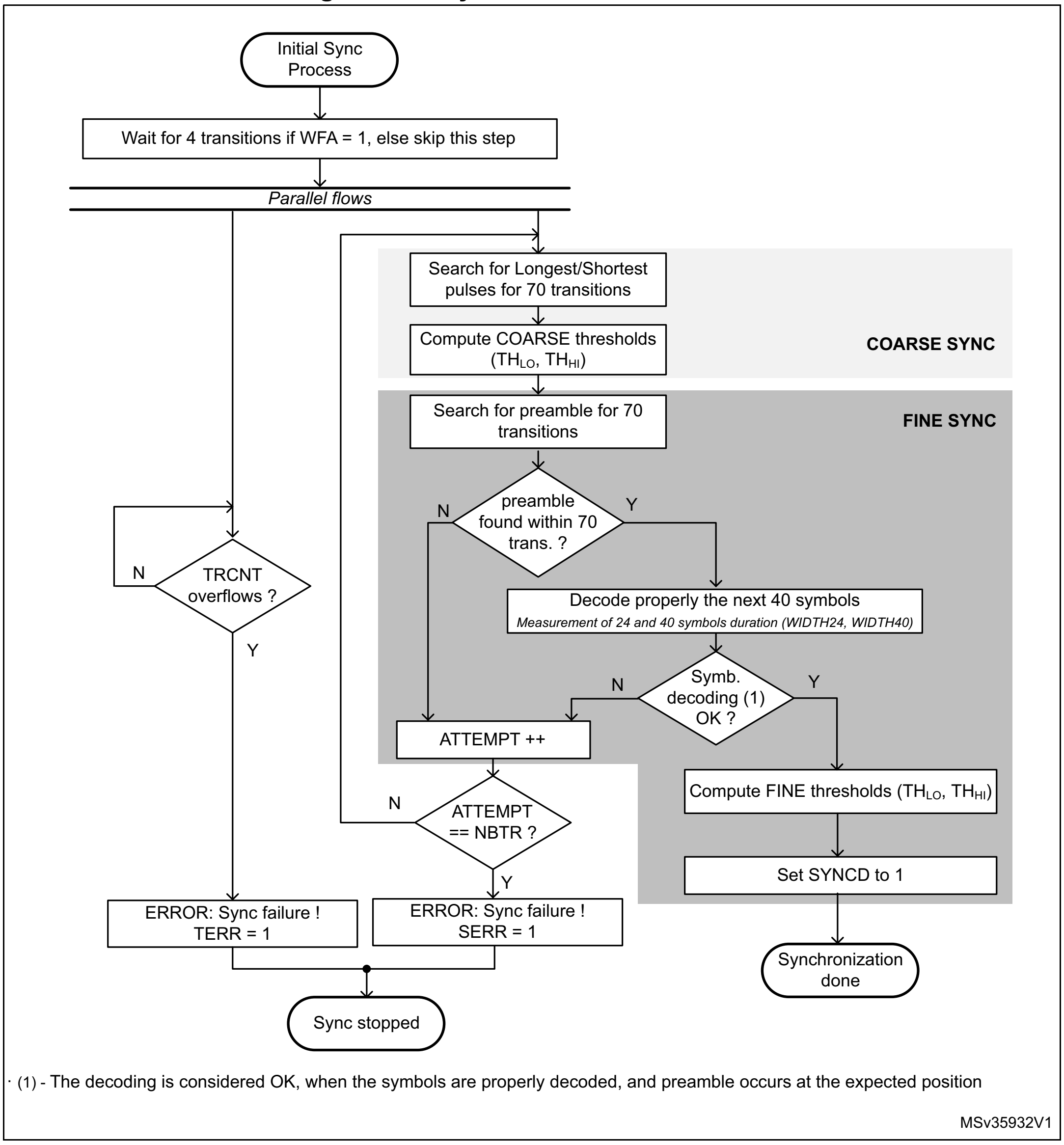
The next step consists on doing a first estimate of the thresholds (COARSE SYNC), in order to perform the fine synchronization (FINE SYNC). Due to disturbances of the SPDIFRX line, it can happen that the process is not executed first time right. For this purpose, the user can program the number of allowed re-tries (NBTR) before setting SERR error flag.
When the SPDIFRX is able to measure properly the duration of 24 and 40 consecutive symbols then the FINE SYNC is completed, the threshold values are updated, and the flag SYNCD is set to 1. Refer to *Section : Transition coder and preamble detector* for additional information.

Two kinds of errors are detected:

- An overflow of the TRCNT, which generally means that there is no valid S/PDIF stream in the input line. This overflow is indicated by TERR flag.
- The number of retries reached the programmed value. This means that strong jitter is present on the S/PDIF signal. This error is indicated by SERR flag.

When the first FINE SYNC is completed, the reception of channel status (C) and user data (U) starts when the next “B” preamble is detected (see *Figure 640*).Then the user can read IEC-60958 C and U bits through SPDIFRX_CSR register. According to this information the user can then select the proper settings for DRFMT and RXSTEO. For example if the user detects that the current audio stream transports encoded data, then he can put RXSTEO to 0, and DRFMT to 10 prior to start data reception. Note that DRFMT and RXSTEO cannot be modified when SPDIFRXEN = 11. Writes to these fields are ignored if SPDIFRXEN is already 11, though these field can be changed with the same write instruction that causes SPDIFRXEN to become 11.
Then the SPDIFRX waits for SPDIFRXEN = 11 and the “B” preamble before starting saving audio samples.

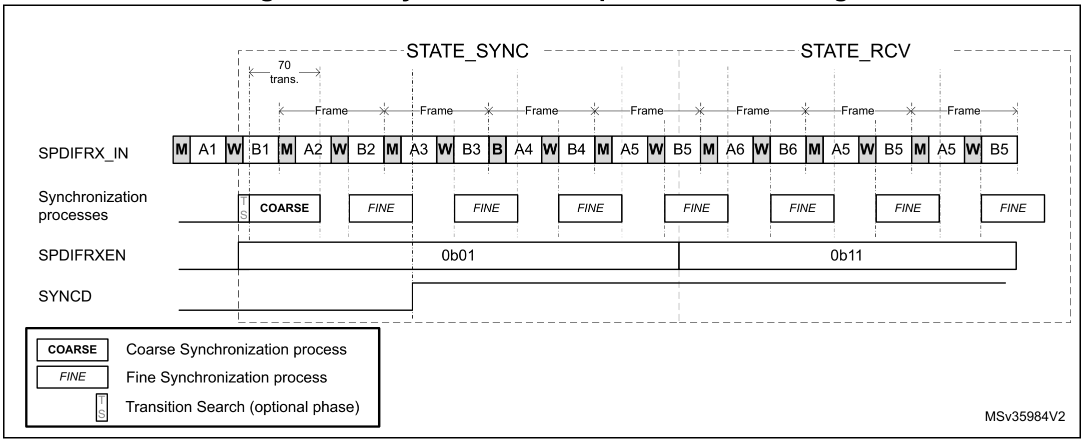
Figure 636. Synchronization flowchart



Refer to *Frame structure and synchronization error* for additional information concerning TRCNT overflow.

The FINE SYNC process is re-triggered every frame in order to update thresholds as shown in *Figure 637* in order to continuously track S/PDIF synchronization.

Figure 637. Synchronization process scheduling



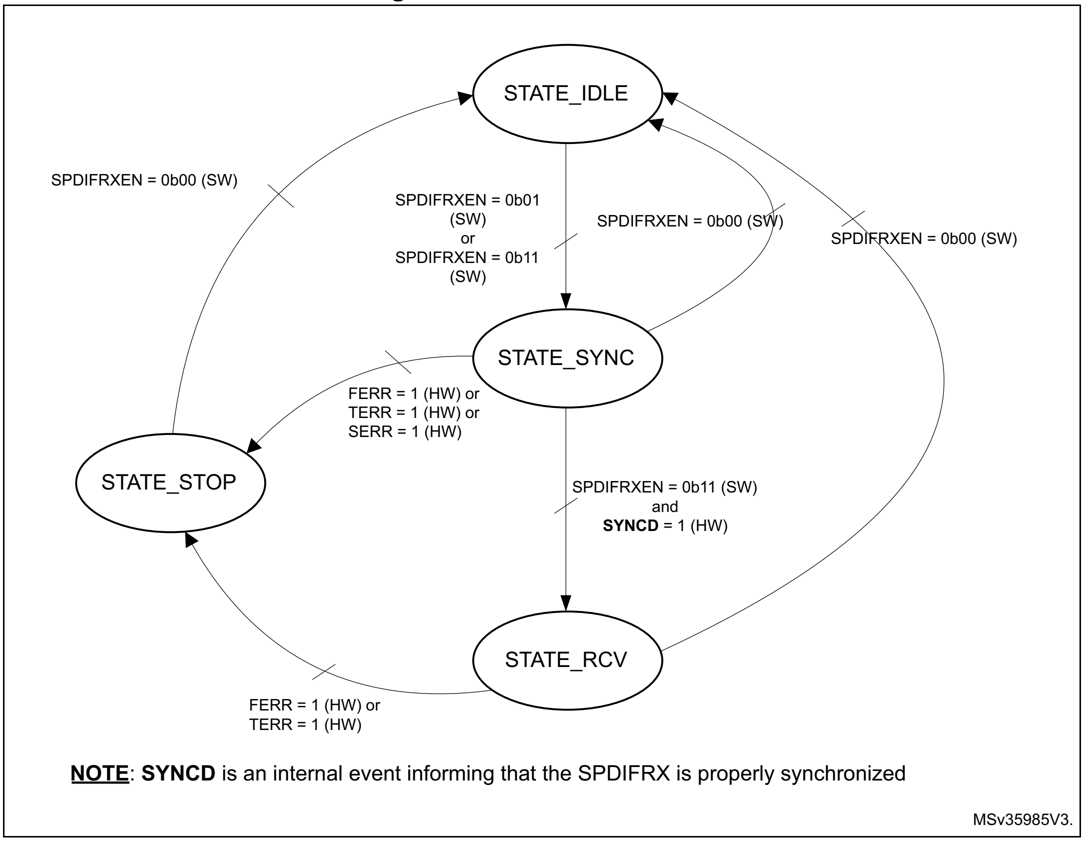
54.3.6 SPDIFRX handling

The software can control the state of the SPDIFRX through SPDIFRXEN field. The SPDIFRX can be into one of the following states:

- STATE_IDLE:
  The peripheral is disabled, the spdifrx_ker_ck domain is reset. The spdifrx_pclk domain is functional.
- STATE_SYNC:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user and channel status can be read via interrupt of DMA. The audio samples are not provided to receive buffer.
- STATE_RCV:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user, channel status and audio samples can be read via interrupt or DMA channels. When SPDIFRXEN goes to 11, the SPDIFRX waits for “B” preamble before starting saving audio samples.
- STOP_STATE:
  The peripheral is no longer synchronized, the reception of the user, channel status and audio samples are stopped. It is expected that the software re-starts the SPDIFRX.

Figure 638 shows the possible states of the SPDIFRX, and how to transition from one state to the other. The bits under software control are followed by the mention “(SW)”, the bits under SPDIFRX control are followed by the mention “(HW)”.

Figure 638. SPDIFRX States



When SPDIFRX is in STATE_IDLE:

- The software can transition to STATE_SYNC by setting SPDIFRXEN to 01 or 11

When SPDIFRX is in STATE_SYNC:

- If the synchronization fails or if the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- When the synchronization phase is completed, if SPDIFRXEN = 01 the peripheral remains in this state.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.
- The SPDIFRX goes to STATE_RCV if SPDIFRXEN = 11 and if the SYNCD = 1

When SPDIFRX is in STATE_RCV:

- If the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.

When SPDIFRX is in STATE_STOP:

- The SPDIFRX stops reception and synchronization, and waits for the software to set the bit SPDIFRXEN to 0, in order to clear the error flags.

When SPDIFRXEN is set to 0, the SPDIFRX is disabled, meaning that all the state machines are reset, and RX_BUF is flushed. Note as well that flags FERR, SERR and TERR are reset.

54.3.7 Data reception management

The SPDIFRX offers a double buffer for the audio sample reception. A 32-bit buffer located into the spdifrx_ker_ck clock domain (RX_BUF), and the SPDIFRX_FMTx_DR register. The valid data contained into the RX_BUF are immediately transferred into SPDIFRX_FMTx_DR if SPDIFRX_FMTx_DR is empty.

The valid data contained into the RX_BUF are transferred into SPDIFRX_FMTx_DR when the two following conditions are reached:

- The transition between the parity bit (P) and the next preamble is detected (this indicated that the word is completely received).
- The SPDIFRX_FMTx_DR is empty.

Having a 2-word buffer gives more flexibility for the latency constraint.

The maximum latency allowed is $T_{SAMPLE} - 2T_{PCLK} - 2T_{spdifrx\_ker\_ck}$

Where T_{SAMPLE} is the audio sampling rate of the received stereo audio samples, T_{PCLK} is the period of spdifrx_pclk clock, and $T_{spdifrx\_ker\_ck}$ is the period of spdifrx_ker_ck clock.

The SPDIFRX offers the possibility to use either DMA (spdifrx_dat_dma and spdifrx_ctrl_dma) or interrupts for transferring the audio samples into the memory. The recommended option is DMA, refer to *Section 54.3.12: DMA interface* for additional information.

The SPDIFRX offers several way on handling the received data. The user can either have a separate flow for control information and audio samples, or get them all together.

For each sub-frame, the data reception register SPDIFRX_FMTx_DR contains the 24 data bits, and optionally the V, U, C, PE status bits, and the PT (see *Mixing data and control flow*).

Note that PE bit stands for parity rrror bit, and is set to 1 when a parity error is detected in the decoded sub-frame.
The PT field carries the preamble type (B, M or W).

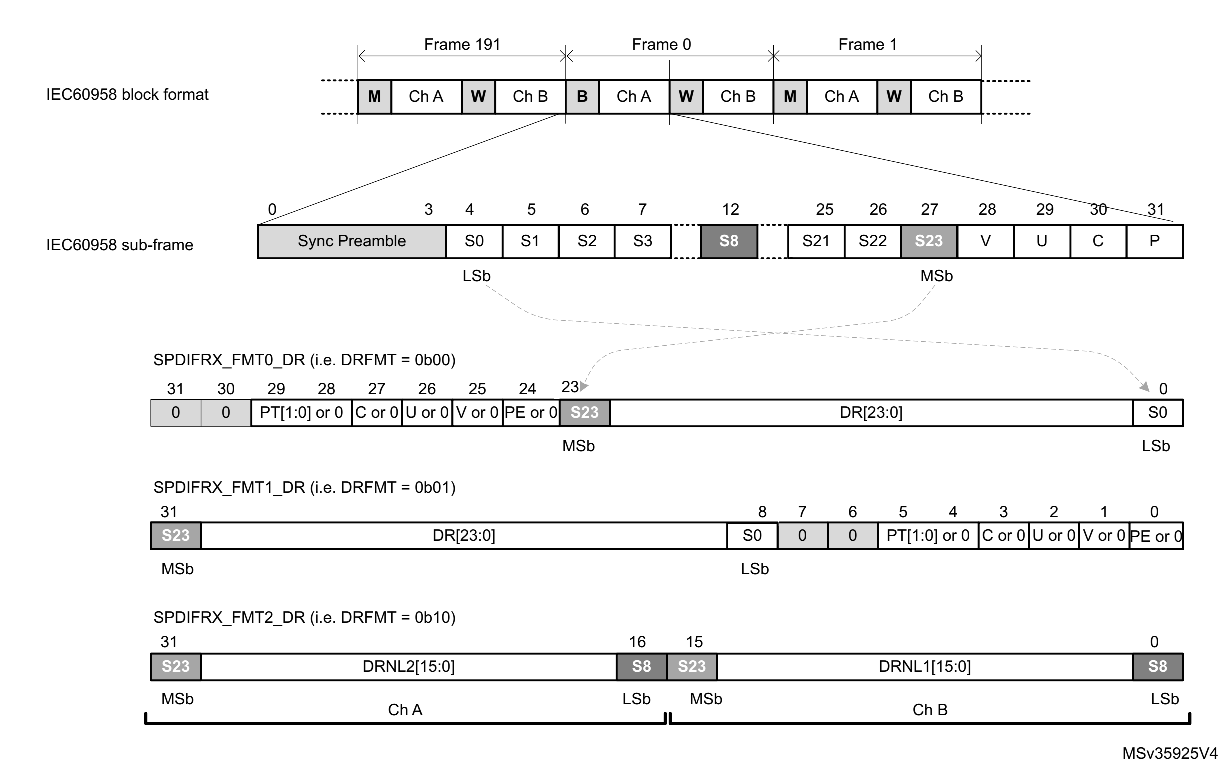
V, U and C are a direct copy of the value received from the S/PDIF interface.

The bit DRFMT allows the selection between 3 audio formats as shown in *Figure 639*.

This document describes 3 data registers: SPDIFRX_FMTx[2:0] (x = 2 to 0), but in reality there is only one physical data register, having 3 possible formats:

- When DRFMT = 0, the format of the data register is the one described by SPDIFRX_FMT0_DR
- When DRFMT = 1, the format of the data register is the one described by SPDIFRX_FMT1_DR
- When DRFMT = 2, the format of the data register is the one described by SPDIFRX_FMT2_DR"

Figure 639. SPDIFRX_FMTx_DR register format



Setting DRFMT to 00 or 01, offers the possibility to have the data either right or left aligned into the SPDIFRX_FMTx_DR register. The status information can be enabled or forced to zero according to the way the software wants to handle them.

The format given by DRFMT= 10 is interesting in non-linear mode, as only 16 bits per sub-frame are used. By using this format, the data of two consecutive sub-frames are stored into SPDIFRX_FMTx_DR, dividing by two the amount of memory footprint. Note that when RXSTEO = 1, there is no misalignment risks (i.e. data from ChA are always stored into SPDIFRX_FMTx_DR[31:16]). If RXSTEO = 0, then there is a misalignment risk is case of overrun situation. In that case SPDIFRX_FMTx_DR[31:16] always contain the oldest value and SPDIFRX_FMTx_DR[15:0] the more recent value (see *Figure 641*).

In this format the status information cannot be mixed with data, but the user can still get them through SPDIFRX_CSR register, and use a dedicated DMA channel or interrupt to transfer them to memory (see *Section 54.3.8: Dedicated control flow*)

Mixing data and control flow

The user can choose to use this mode in order to get the full flexibility of the handling of the control flow. The user can select which field must be kept into the data register (SPDIFRX_FMTx_DR).

- When bit PMSK = 1, the parity error information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit VMSK = 1, the validity information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

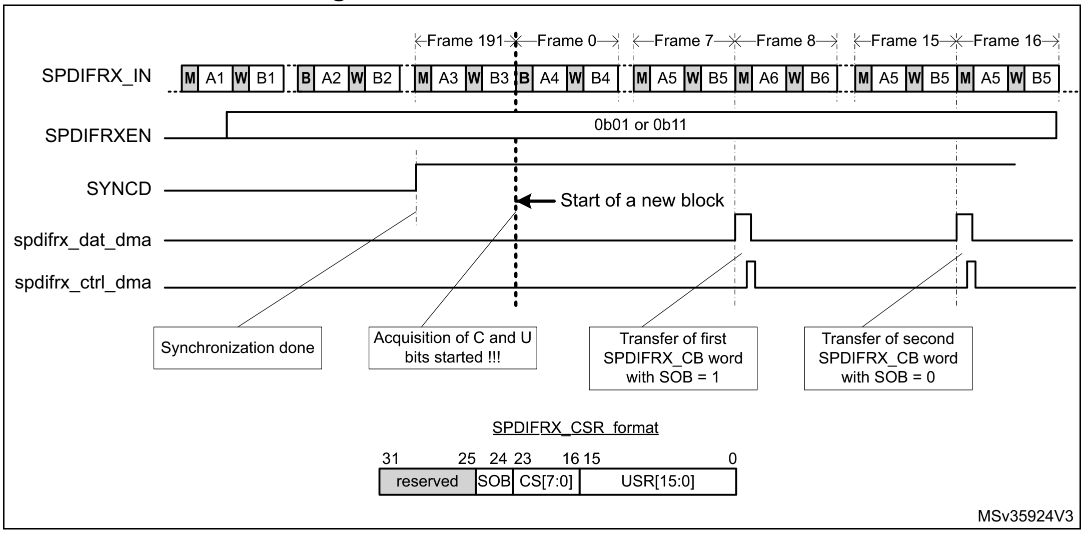
- When bit CUMSK = 1, the channel status, and used data information are masked (set to 0), otherwise they are copied into SPDIFRX_FMTx_DR.
- When bit PTMSK = 1, the preamble type is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

54.3.8 Dedicated control flow

The SPDIFRX offers the possibility to catch both user data and channel status information via a dedicated DMA channel. This feature allows the SPDIFRX to acquire continuously the channel status and user information. The acquisition starts at the beginning of a IEC 60958 block. Two fields are available to control this path: CBDMAEN and SPDIFRXEN. When SPDIFRXEN is set to 01 or 0x11, the acquisition is started, after completion of the synchronization phase. When 8 channel status and 16 user data bits are received, they are packed and stored into SPDIFRX_CSR register. A DMA request is triggered if the bit CBDMAEN is set to 1 (see *Figure 640*).

If CS[0] corresponds to the first bit of a new block, the bit SOB is set to 1. Refer to *Section 54.5.8: SPDIFRX channel status register (SPDIFRX_CSR)*. A bit is available (CHSEL) in order to select if the user wants to select channel status information (C) from the channel A or B.

Figure 640. Channel/user data format



Note: Once the first start of block is detected (B preamble), the SPDIFRX is checking the preamble type every 8 frames.

Note: Overrun error on SPDIFRX_FMTx_DR register does not affect this path.

54.3.9 Reception errors

Frame structure and synchronization error

The SPDIFRX, detects errors, when one of the following condition occurs:

- The FERR bit is set to 1 on the following conditions:
  - For each of the 28 information bits, if one symbol transition sequence is not correct: for example if short pulses are not grouped by pairs.
  - If preambles occur to an unexpected place, or an expected preamble is not received.
- The SERR bit is set when the synchronization fails, because the number of re-tries exceeded the programmed value.
- The TERR bit is set when the counter used to estimate the width between two transitions overflows (TRCNT).
  The overflow occurs when no transition is detected during 8192 periods of spdifrx_ker_ck clock. It represents at most a time interval of 11.6 frames.

When one of those flags goes to 1, the traffic on selected SPDIFRX_IN is then ignored, an interrupt is generated if the IFEIE bit of the SPDIFRX_CR register is set.

The normal procedure when one of those errors occur is:

- Set SPDIFRXEN to 0 in order to clear the error flags
- Set SPDIFRXEN to 01 or 11 in order to restart the SPDIFRX

Refer to *Figure 638* for additional information.

Parity error

For each sub-frame, an even number of zeros and ones is expected inside the 28 information bits. If not, the parity error bit PERR is set in the SPDIFRX_SR register and an interrupt is generated if the parity interrupt enable PERRIE bit is set in the SPDIFRX_CR register. The reception of the incoming data is not paused, and the SPDIFRX continue to deliver data to SPDIFRX_FMTx_DR even if the interrupt is still pending.

The interrupt is acknowledged by clearing the PERR flag through PERRCF bit.

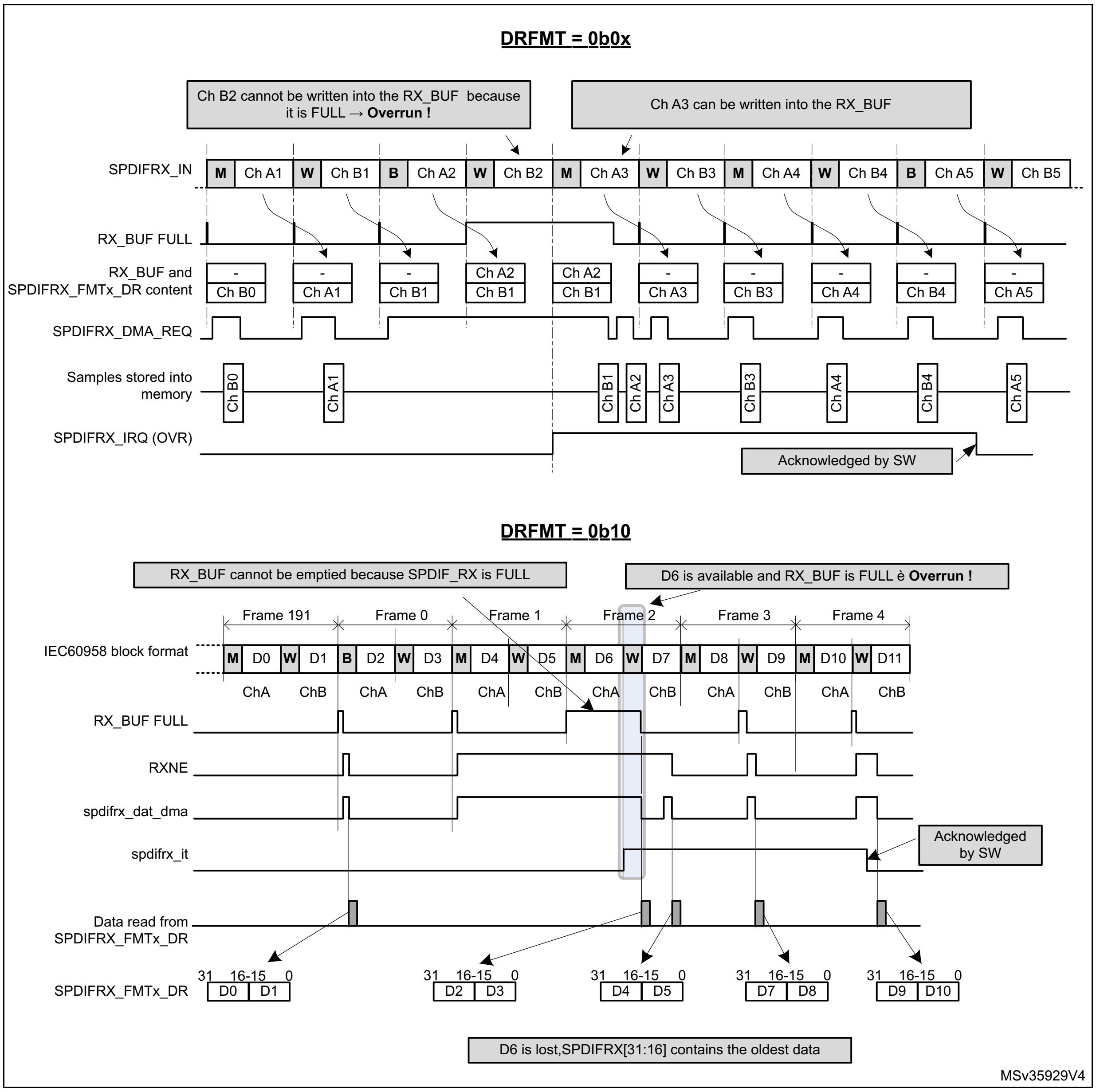
If the software wants to guarantee the coherency between the data read in the SPDIFRX_FMTx_DR register and the value of the bit PERR, the bit PMSK must be set to 0.

Overrun error

If both SPDIFRX_FMTx_DR and RX_BUF are full, while the SPDIFRX_DC needs to write a new sample in RX_BUF, this new sample is dropped, and an overrun condition is triggered. The overrun error flag OVR is set in the SPDIFRX_SR register and an interrupt is generated if the OVRIE bit of the SPDIFRX_CR register is set.

If the RXSTEO bit is set to 0, then as soon as the RX_BUF is empty, the SPDIFRX stores the next incoming data, even if the OVR flag is still pending. The main purpose is to reduce as much as possible the amount of lost samples. Note that the behavior is similar independently of DRFMT value. See *Figure 641*.

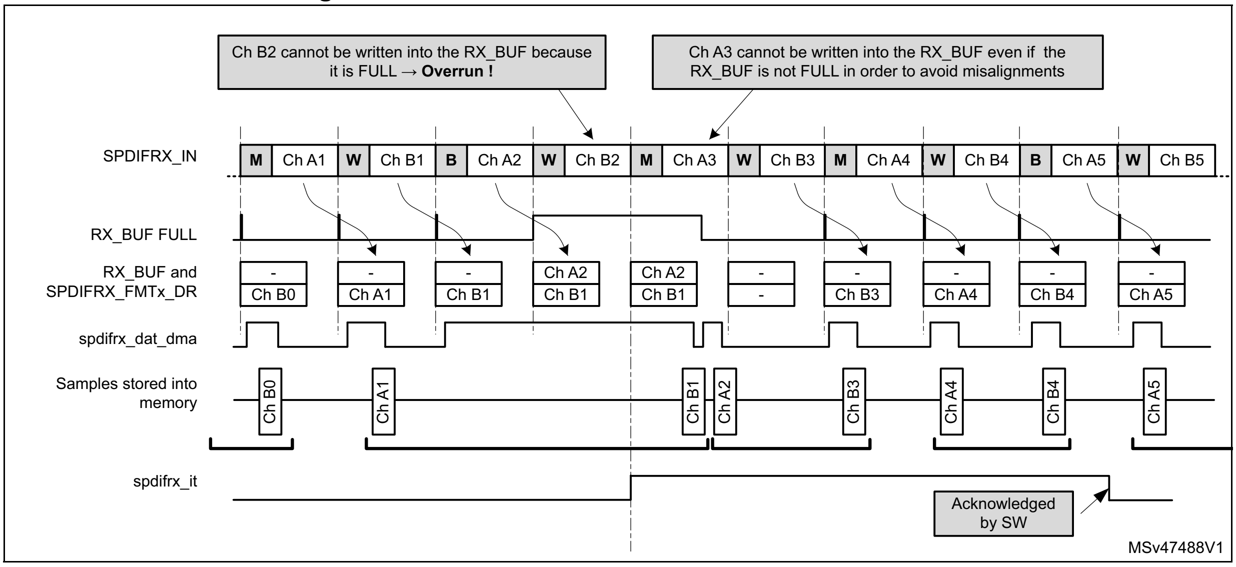
Figure 641. S/PDIF overrun error when RXSTEO = 0



If the RXSTEO bit is set to 1, it means that stereo data are transported, then the SPDIFRX has to avoid misalignment between left and right channels. So the peripheral has to drop a second sample even if there is room inside the RX_BUF in order to avoid misalignment. Then the incoming samples can be written normally into the RX_BUF even if the OVR flag is still pending. Refer to *Figure 642*.

The OVR flag is cleared by software, by setting the OVRCF bit to 1.

Figure 642. S/PDIF overrun error when RXSTEO = 1



54.3.10 Clocking strategy

The SPDIFRX block needs two different clocks:

- The APB clock (spdifrx_pclk), which is used for the register interface,
- The spdifrx_ker_ck which is mainly used by the SPDIFRX_DC part. Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized (SYNC block on *Figure 628*).

In order to decode properly the incoming S/PDIF stream the SPDIFRX_DC must re-sample the received data with a clock at least 11 times higher than the maximum symbol rate, or 704 times higher than the audio sample rate. For example if the user expects to receive a symbol rate to up to 12.288 MHz, the sample rate must be at least 135.2 MHz. The clock used by the SPDIFRX_DC is the spdifrx_ker_ck.

The frequency of the spdifrx_pclk must be at least equal to the symbol rate.

Table 397. Minimum spdifrx_ker_ck frequency versus audio sampling rate(1)

Symbol rate	Minimum spdifrx_ker_ck frequency	Comments
3.072 MHz	33.8 MHz	For 48 kHz stream
6.144 MHz	67.6 MHz	For 96 kHz stream
12.288 MHz	135.2 MHz	For 192 kHz stream

1. Check the RCC capabilities in order to verify which sampling rates can be supported.

54.3.11 Symbol clock generation

The SPDIFRX block provides a symbol clock on signal named **spdifrx_symb_ck**, which can be used as the reference kernel clock for another audio device such as SAI or SPI/I2S. It can be used for SPDIFRX to I2S bridge function.

The symbol clock is built using the values of WIDTH24, WIDTH40 and the symbol boundaries.

- During the reception of the sub-frame sync preambles, the falling and rising edges of the symbol clock are built from the WIDTH24 and WIDTH40 values. Note that WIDTH24 and WIDTH40 are also used for the generation of the symbol clock, when the SPDIFRX is STATE_STOP or STATE_IDLE. See *Table 398* for details.
- During the reception of the sub-frame payload, the SPDIFRX uses the symbols boundaries to generate the rising edge, the WIDTH24 and WIDTH40 values for the generation of the falling edge.

The duty cycle of the symbol clock is close to 50% during the reception of the sub-frame payload. However, the duty cycle can be altered when the SPDIFRX transitions from a symbol clock generated with WIDTH24 and WIDTH40 to a clock generated by the symbol clock boundaries or vice-versa.

The symbol clock has an important jitter mainly due to:

- The re-sampling of the S/PDIF signal with spdifrx_ker_ck clock
- The transition of the symbol clock generation mode

For that reason the application must consider the quality degradation if the symbol clock is used as the reference clock for A/D or D/A converters.

The generation of this symbol clock is controlled by the CKSEN bit. When CKSEN = ‘1’, the clock symbol is generated when the SPDIFRX completes successfully the first fine synchronization (SYNCD = 1), and when it receives correct data from the selected SPDIFRX input.

When the SPDIFRX goes to STATE_STOP, or STATE_IDLE, the symbol clock is gated if the bit CKSBKPEN = ‘0’. If the CKSBKPEN = ‘1’, then a backup symbol clock is still generated if the SPDIFRX is properly synchronized (i.e. valid values available for WIDTH24 and WIDTH40). *Table 398* gives more details on the conditions controlling the generation of the symbol clock.

Table 398. Conditions of spdifrx_symb_ck generation

<table>
<tr><th>SPDIFRX states and conditions</th><th>CKSEN</th><th>CKSBKPEN</th><th>spdifrx_symb_ck state</th></tr>
<tr><td>Any state</td><td>0</td><td>X</td><td>Disabled</td></tr>
<tr><td>– SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or,
– SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input.</td><td rowspan="2">1</td><td>0</td><td>Enabled</td></tr>
<tr><td>– SPDIFRX in STATE_IDLE or,
– SPDIFRX in STATE_STOP or,
– SPDIFRX did not complete the fine synchronization (on-going)
– SPDIFRX is in STATE_RCV, but no data (transitions) detected on the selected SPDIFRX input.</td><td>0</td><td>Disabled</td></tr>
</table>

Table 398. Conditions of spdifrx_symb_ck generation (continued)

<table>
<tr><th>SPDIFRX states and conditions</th><th>CKSEN</th><th>CKSBKPEN</th><th>spdifrx_symb_ck state</th></tr>
<tr><td>– SPDIFRX in STATE_IDLE, but with valid values for WIDTH40 and WIDTH24 or
– SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or,
– SPDIFRX in STATE_SYNC the on-going fine synchronization is not completed, but WIDTH40 and WIDTH24 contain the valid values from the previous synchronization or,
– SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input or,
– SPDIFRX in STATE_STOP, but with valid values for WIDTH40 and WIDTH24.</td><td rowspan="2">1</td><td rowspan="2">1</td><td>Enabled</td></tr>
<tr><td>– SPDIFRX in IDLE, with invalid values for WIDTH40 and WIDTH24 or,
– SPDIFRX in STOP with invalid values for WIDTH40 and WIDTH24 (SERR = '1') or,
– SPDIFRX in STATE_SYNC with invalid values for WIDTH40 and WIDTH24, and did not completed the on-going fine synchronization or,
– SPDIFRX in STATE_RCV and no transitions detected on the selected SPDIFRX input</td><td>Disabled</td></tr>
</table>

Note that when the flag SERR is set to ‘1’, neither the symbol clock nor the backup clock can be generated, since there is no synchronization.

Note that when both CKSEN and CKSBKPEN are set to ‘1’, the symbol clock looses some transitions when the SPDIFRX switches from STATE_SYNC or STATE_RCV to STATE_STOP, or STATE_IDLE.

The bits CKSEN and CKSBKPEN are located into *SPDIFRX control register (SPDIFRX_CR)*.

54.3.12 DMA interface

The SPDIFRX interface is able to perform communication using the DMA.

Note: The user must refer to product specifications for availability of the DMA controller.

The SPDIFRX offers two independent DMA channels:

- A DMA channel dedicated to the data transfer
- A DMA channel dedicated to the channel status and user data transfer

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register without CPU intervention.

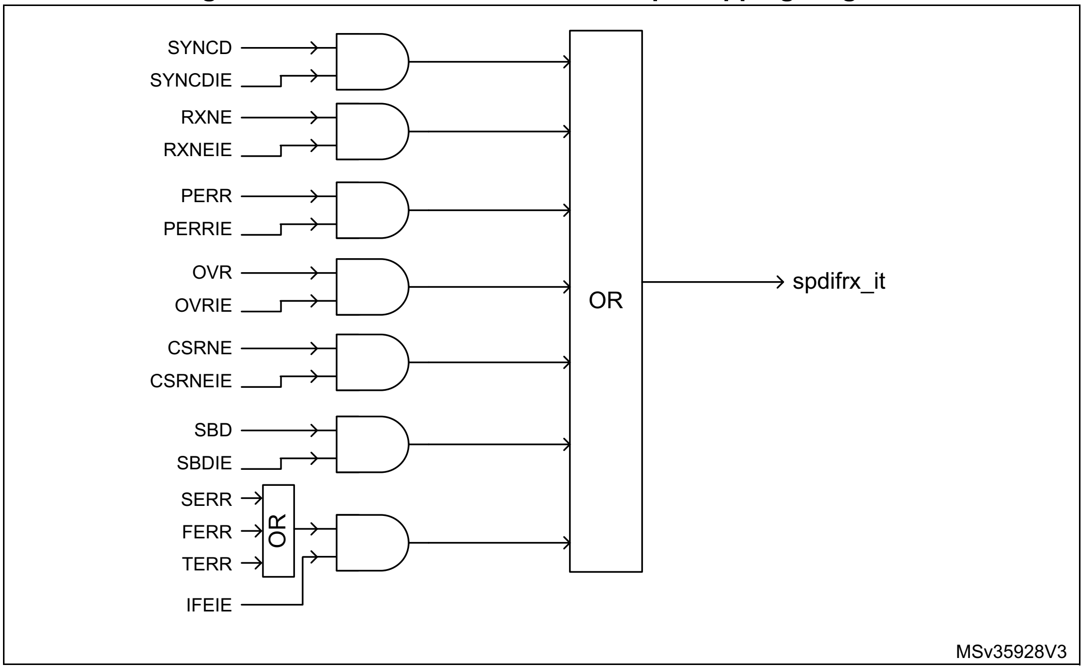
For the use of DMA for the control data refer to *Section 54.3.8: Dedicated control flow*.

54.3.13 Interrupt generation

An interrupt line is shared between:

- Reception events for data flow (RXNE)
- Reception event for control flow (CSRNE)
- Data corruption detection (PERR)
- Transfer flow interruption (OVR)
- Frame structure and synchronization errors (SERR, TERR and FERR)
- Start of new block interrupt (SBD)
- Synchronization done (SYNCD)

Figure 643. SPDIFRX interface interrupt mapping diagram



Clearing interrupt source

- RXNE is cleared when SPDIFRX_FMTx_DR register is read
- CSRNE is cleared when SPDIFRX_CSR register is read
- FERR is cleared when SPDIFRXEN is set to 0
- SERR is cleared when SPDIFRXEN is set to 0
- TERR is cleared when SPDIFRXEN is set to 0
- Others are cleared through SPDIFRX_IFCR register

Note: The SBD event can only occur when the SPDIFRX is synchronized to the input stream (SYNCD = 1).
The SBD flag behavior is not guaranteed when the sub-frame which contains the B preamble is lost due to an overrun.

54.3.14 Register protection

The SPDIFRX block embeds some hardware protection avoid erroneous use of control registers. The table hereafter shows the bit field properties according to the SPDIFRX state.

Table 399. Bit field property versus SPDIFRX state

Registers	Field	SPDIFRXEN		
		00 (STATE_IDLE)	01 (STATE_SYNC)	11 (STATE_RCV)
SPDIFRX_CR	INSEL	rw	r	r
	WFA	rw	r	r
	NBTR	rw	r	r
	CHSEL	rw	r	r
	CBDMAEN	rw	rw	rw
	PTMSK	rw	rw	rw
	CUMSK	rw	rw	rw
	VMSK	rw	rw	rw
	PMSK	rw	rw	rw
	DRFMT	rw	rw	r
	RXSTEO	rw	rw	r
	RXDMAEN	rw	rw	rw
SPDIFRX_IMR	All fields	rw	rw	rw

The table clearly shows that fields such as INSEL must be programmed when the SPDIFRX is in STATE_IDLE. In the others SPDIFRX states, the hardware prevents writing to this field.

Note: Even if the hardware allows the writing of CBDMAEN and RXDMAEN "on-the-fly", it is not recommended to enable the DMA when the SPDIFRX already receives data.

Each of the mask bits (such as PMSK, VMSK) can be changed "on-the-fly" at any SPDIFRX state, but any change does not affect data which are already hold in SPDIFRX_FMTx_DR.

54.4 Programming procedures

The following example illustrates a complete activation sequence of the SPDIFRX block. The data path and channel status and user information both use a dedicated DMA channel. The activation sequence is then split into the following steps:

- Wait for valid data on the selected SPDIFRX_IN input
- Synchronize to the S/PDIF stream
- Read the channel status and user information in order to setup the complete audio path
- Start data acquisition

A simple way to check if valid data are available into the SPDIFRX_IN line is to switch the SPDIFRX into the STATE_SYNC, with bit WFA set to 1. The description hereafter focuses on detection. It is also possible to implement this function as follows:

- The software has to check from time to time (i.e. every 100 ms for example) if the SPDIFRX can find synchronization. This can be done by checking if the bit TERR is set. When it is set it indicates that no activity as been found.
- Connect the SPDIFRX_IN input to an external interrupt event block in order to detect transitions of SPDIFRX_IN line. When activity is detected, then SPDIFRXEN can be set to 01 or 11.

  For those two implementations, the bit WFA is set to 0.

54.4.1 Initialization phase

- The initialization function looks like this:
- Configure the DMA transfer for both audio samples and IEC60958 channel status and user information (DMA channel selection and activation, priority, number of data to transfer, circular/no circular mode, DMA interrupts)
- Configure the destination address:
  - Configure the address of the SPDIFRX_CSR register as source address for IEC60958 channel status and user information
  - Configure the address of the SPDIFRX_FMTx_DR register as source address for audio samples
  - Enable the generation of the spdifrx_ker_ck. Refer to *Table 397* in order to define the minimum clock frequency versus supported audio sampling rate.
    Note that the audio sampling rate of the received stream is not known in advance. This means that the user has to select a spdifrx_ker_ck frequency at least 704 times higher than the maximum audio sampling rate the application is supposed to handle: for example if the application is able to handle streams to up to 96 kHz, then $F_{spdifrx\_ker\_ck}$ must be at least 704 x 96 kHz = 67.6 MHz
- Enable interrupt for errors and event signaling (IFEIE = SYNCDIE = OVRIE, PERRIE = 1, others set to 0). Note that SYNCDIE can be set to 0.
- Configure the SPDIFRX_CR register:
  - INSEL must select the wanted input
  - NBTR = 2, WFA = 1 (16 re-tries allowed, wait for activity before going to synchronization phase),
  - PTMSK = CUMSK = 1 (Preamble, C and U bits are not mixed with data)
  - VMSK = PMSK = 0 (Parity error and validity bit mixed with data)
  - CHSEL = 0 (channels status are read from sub-frame A)
  - DRFMT = 01 (data aligned to the left)
  - RXSTEO = 1 (expected stereo mode linear)
  - CBDMAEN = RXDMAEN = 1 (enable DMA channels)
  - SPDIFRXEN = 01 (switch SPDIFRX to STATE_SYNC)
- The CPU can enter in WFI mode

Then the CPU receives interrupts coming either from DMA or SPDIFRX.

54.4.2 Handling of interrupts coming from SPDIFRX

When an interrupt from the SPDIFRX is received, then the software has to check what is the source of the interrupt by reading the SPDIFRX_SR register.

- If SYNCD is set to 1, then it means that the synchronization is properly completed. No action has to be performed in our case as the DMA is already programmed. The software just needs to wait for DMA interrupt in order to read channel status information.
  The SYNCD flag must be cleared by setting SYNCDCF bit of SPDIFRX_IFCR register to 1.
- If TERR or SERR or FERR are set to 1, the software has to set SPDIFRXEN to 0, and re-start from the initialization phase.
  - TERR indicates that a time-out occurs either during synchronization phase or after.
  - SERR indicates that the synchronization fails because the maximum allowed re-tries are reached.
  - FERR indicates that the reading of information after synchronization fails (such as unexpected preamble, bad data decoding).
- If PERR is set to 1, it means that a parity error is detected, so one of the received audio sample or the channel status or user data bits are corrupted. The action taken here depends on the application: one action can be to drop the current channel status block as it is not reliable. There is no need to re-start from the initialization phase, as the synchronization is not lost.
  The PERR flag must be cleared by setting PERRCF bit of SPDIFRX_IFCR register to 1.

54.4.3 Handling of interrupts coming from DMA

If an interrupt comes from the DMA channel used of the channel status (SPDIFRX_CSR):

If no error occurred (that is PERR), the CPU can start the decoding of channel information. For example bit 1 of the channel status informs the user if the current stream is linear or not. This information is very important in order to set-up the proper processing chain. In the same way, bits 24 to 27 of the channel status give the sampling frequency of the stream incoming stream.

Thanks to that information, the user can then configure the RXSTEO bit and DRFMT field prior to start the data reception. For example if the current stream is non linear PCM then RXSTEO is set to 0, and DRFMT is set to 10. Then the user can enable the data reception by setting SPDIFRXEN to 11.

The SOB bit, when set to 1 indicates the start of a new block. This information helps the software to identify the bit 0 of the channel status. Note that if the DMA generates an interrupt every time 24 values are transferred into the memory, then the first word always corresponds to the start of a new block.

If an interrupt comes from the DMA channel used of the audio samples (SPDIFRX_FMTx_DR):

The process performed here depends of the data type (linear or non-linear), and on the data format selected.

For example in linear mode, if PE or V bit is set a special processing can be performed locally in order to avoid spurs on output. In non-linear mode those bits are not important as data frame have their own checksum.

54.5 SPDIFRX interface registers

54.5.1 SPDIFRX control register (SPDIFRX_CR)

Only 32-bit accesses are allowed in this register.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKSBKPEN	CKSEN	Res.	INSEL[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CKSBKPEN**: Backup symbol clock enable
This bit is set/reset by software.
1: The SPDIFRX generates a backup symbol clock if CKSEN = 1.
0: The SPDIFRX does not generate a backup symbol clock.

Bit 20 **CKSEN**: Symbol clock enable
This bit is set/reset by software.
1: The SPDIFRX generates a symbol clock.
0: The SPDIFRX does not generate a symbol clock.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **INSEL[2:0]**: SPDIFRX input selection(1)
000: SPDIFRX_IN0 selected
001: SPDIFRX_IN1 selected
010: SPDIFRX_IN2 selected
011: SPDIFRX_IN3 selected
other: reserved

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WFA**: Wait for activity(1)
This bit is set/reset by software.
1: The SPDIFRX waits for activity on SPDIFRX_IN line (4 transitions) before performing the synchronization.
0: The SPDIFRX does not wait for activity on SPDIFRX_IN line before performing the synchronization.

Bits 13:12 **NBTR[1:0]**: Maximum allowed re-tries during synchronization phase(1)
00: No re-try is allowed (only one attempt)
01: 3 re-tries allowed
10: 15 re-tries allowed
11: 63 re-tries allowed

Bit 11 **CHSEL**: Channel selection(1)
This bit is set/reset by software.
1: The control flow takes the channel status from channel B.
0: The control flow takes the channel status from channel A.

Bit 10 **CBDMAEN**: Control buffer DMA enable for control flow(1)
This bit is set/reset by software.
1: DMA mode is enabled for reception of channel status and used data information.
0: DMA mode is disabled for reception of channel status and used data information.
Note: When this bit is set, the DMA request is made whenever the CSRNE flag is set.

Bit 9 **PTMSK**: Mask of preamble type bits(1)
This bit is set/reset by software.
1: The preamble type bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
0: The preamble type bits are copied into the SPDIFRX_FMTx_DR.

Bit 8 **CUMSK**: Mask of channel status and user bits(1)
This bit is set/reset by software.
1: The channel status and user bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
0: The channel status and user bits are copied into the SPDIFRX_FMTx_DR.

Bit 7 **VMSK**: Mask of validity bit(1)
This bit is set/reset by software.
1: The validity bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
0: The validity bit is copied into the SPDIFRX_FMTx_DR.

Bit 6 **PMSK**: Mask parity error bit(1)
This bit is set/reset by software.
1: The parity error bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
0: The parity error bit is copied into the SPDIFRX_FMTx_DR.

Bits 5:4 **DRFMT[1:0]**: RX data format(1)
This bit is set/reset by software.
11: reserved
10: Data sample are packed by setting two 16-bit sample into a 32-bit word.
01: Data samples are aligned in the left (MSB)
00: Data samples are aligned in the right (LSB).

Bit 3 **RXSTEO**: Stereo mode(1)

This bit is set/reset by software.

1: The peripheral is in stereo mode.

0: The peripheral is in mono mode.

Note: This bit is used in case of overrun situation in order to handle misalignment.

Bit 2 **RXDMAEN**: Receiver DMA enable for data flow(1)

This bit is set/reset by software.

1: DMA mode is enabled for reception.

0: DMA mode is disabled for reception.

Note: When this bit is set, the DMA request is made whenever the RXNE flag is set.

Bits 1:0 **SPDIFRXEN[1:0]**: Peripheral block enable(1)

This field is modified by software.

It must be used to change the peripheral phase among the three possible states: STATE_IDLE, STATE_SYNC and STATE_RCV.

00: Disable SPDIFRX (STATE_IDLE).

01: Enable SPDIFRX synchronization only.

10: Reserved

11: Enable SPDIF receiver.

Note: It is not possible to transition from STATE_RCV to STATE_SYNC, the user must first go the STATE_IDLE.
it is possible to transition from STATE_IDLE to STATE_RCV: in that case the peripheral transitions from STATE_IDLE to STATE_SYNC and as soon as the synchronization is performed goes to STATE_RCV.

1. Refer to *Section 54.3.14: Register protection* for additional information on fields properties.

54.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFE IE	SYNCD IE	SBLK IE	OVR IE	PERR IE	CSRNE IE	RXNE IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **IFEIE**: Serial interface error interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited.

1: A SPDIFRX interface interrupt is generated whenever SERR = 1, TERR = 1, or FERR = 1 in the SPDIFRX_SR register.

Bit 5 **SYNCDIE**: Synchronization done

This bit is set and cleared by software.

0: Interrupt is inhibited.

1: A SPDIFRX interface interrupt is generated whenever SYNCD = 1 in the SPDIFRX_SR register.

Bit 4 **SBLKIE**: Synchronization block detected interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever SBD = 1 in the SPDIFRX_SR register.

Bit 3 **OVRIE**: Overrun error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever OVR=1 in the SPDIFRX_SR register.

Bit 2 **PERRIE**: Parity error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever PERR=1 in the SPDIFRX_SR register.

Bit 1 **CSRNEIE:** Control buffer ready interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever CSRNE = 1 in the SPDIFRX_SR register.

Bit 0 **RXNEIE**: RXNE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever RXNE=1 in the SPDIFRX_SR register.

54.5.3 SPDIFRX status register (SPDIFRX_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WIDTH5[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
							r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **WIDTH5[14:0]**: duration of 5 symbols counted with spdifrx_ker_ck
This value represents the amount of spdifrx_ker_ck clock periods contained on a length of 5 consecutive symbols. This value can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of spdifrx_ker_ck.
For example if the spdifrx_ker_ck is fixed to 84 MHz, and WIDTH5 = 147d. The estimated sampling rate of the S/PDIF stream is:
$Fs = 5 \times F_{spdifrx\_ker\_ck} / (WIDTH5 \times 64) \sim 44.6$ kHz, so the closest standard sampling rate is 44.1 kHz.
Note that WIDTH5 is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TERR**: Time-out error

This bit is set by hardware when the counter TRCNT reaches its max value. It indicates that the time interval between two transitions is too long. It generally indicates that there is no valid signal on SPDIFRX_IN input.

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No sequence error is detected.

1: Sequence error is detected.

Bit 7 **SERR**: Synchronization error

This bit is set by hardware when the synchronization fails due to amount of re-tries for NBTR.

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE = 1 in the SPDIFRX_IMR register.

0: No synchronization error is detected.

1: Synchronization error is detected.

Bit 6 **FERR**: Framing error

This bit is set by hardware when an error occurs during data reception: such as preamble not at the expected place, short transition not grouped by pairs.

This is set by the hardware only if the synchronization is completed (SYNCD = 1).

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No Manchester violation detected

1: Manchester violation detected

Bit 5 **SYNCD**: Synchronization done

This bit is set by hardware when the initial synchronization phase is properly completed.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if SYNCDIE = 1 in the SPDIFRX_IMR register.

0: Synchronization is pending.

1: Synchronization is completed.

Bit 4 **SBD**: Synchronization block detected

This bit is set by hardware when a “B” preamble is detected.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if SBLKIE = 1 in the SPDIFRX_IMR register.

0: No “B” preamble is detected.

1: “B” preamble is detected.

Bit 3 **OVR**: Overrun error

This bit is set by hardware when a received data is ready to be transferred in the SPDIFRX_FMTx_DR register while RXNE = 1 and both SPDIFRX_FMTx_DR and RX_BUF are full.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if OVRIE=1 in the SPDIFRX_IMR register.

0: No overrun error

1: Overrun error is detected.

Note: When this bit is set, the SPDIFRX_FMTx_DR register content is not lost but the last data received are.

Bit 2 **PERR**: Parity error

This bit is set by hardware when the data and status bits of the sub-frame received contain an odd number of 0 and 1.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if PIE = 1 in the SPDIFRX_IMR register.

0: No parity error

1: Parity error

Bit 1 **CSRNE**: Control buffer register not empty

This bit is set by hardware when a valid control information is ready.

This flag is cleared when reading SPDIFRX_CSR register.

An interrupt is generated if CBRDYIE = 1 in the SPDIFRX_IMR register.

0: No control word available on SPDIFRX_CSR register

1: A control word is available on SPDIFRX_CSR register.

Bit 0 **RXNE**: Read data register not empty

This bit is set by hardware when a valid data is available into SPDIFRX_FMTx_DR register.

This flag is cleared by reading the SPDIFRX_FMTx_DR register.

An interrupt is generated if RXNEIE=1 in the SPDIFRX_IMR register.

0: Data is not received.

1: Received data is ready to be read.

54.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCD CF	SBD CF	OVR CF	PERR CF	Res.	Res.
										w	w	w	w		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **SYNCDCF**: clears the synchronization done flag

Writing 1 in this bit clears the flag SYNCD in the SPDIFRX_SR register.

Reading this bit always returns the value 0.

Bit 4 **SBDCF**: clears the synchronization block detected flag

Writing 1 in this bit clears the flag SBD in the SPDIFRX_SR register.

Reading this bit always returns the value 0.

Bit 3 **OVRCF**: clears the overrun error flag

Writing 1 in this bit clears the flag OVR in the SPDIFRX_SR register.

Reading this bit always returns the value 0.

Bit 2 **PERRCF**: clears the parity error flag

Writing 1 in this bit clears the flag PERR in the SPDIFRX_SR register.

Reading this bit always returns the value 0.

Bits 1:0 Reserved, must be kept at reset value.

54.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 00:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:16]							
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **PT[1:0]:** preamble type
These bits indicate the preamble received.
00: not used
01: Preamble B received
10: Preamble M received
11: Preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 27 **C**: channel status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 26 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 25 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 24 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

Bits 23:0 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

54.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 01:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DR[7:0]								Res.	Res.	PT[1:0]		C	U	V	PE
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:8 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PT[1:0]:** preamble type
These bits indicate the preamble received.
00: not used
01: preamble B received
10: preamble M received
11: preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 3 **C**: channel Status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 2 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 1 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 0 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

54.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT.

The data format proposed when DRFMT = 10, is dedicated to non-linear mode, as only 16 bits are used (bits 23 to 8 from S/PDIF sub-frame).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DRNL2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRNL1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **DRNL2[15:0]**: data value
This field contains the channel A

Bits 15:0 **DRNL1[15:0]**: data value
This field contains the channel B

54.5.8 SPDIFRX channel status register (SPDIFRX_CSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]							
							r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SOB**: start of block

This bit indicates if the bit CS[0] corresponds to the first bit of a new block

0: CS[0] is not the first bit of a new block

1: CS[0] is the first bit of a new block

Bits 23:16 **CS[7:0]**: channel A status information

Bit CS[0] is the oldest value

Bits 15:0 **USR[15:0]**: user data information

Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B.

So USR[n] bits come from channel A is n is even, otherwise they come from channel B.

54.5.9 SPDIFRX debug information register (SPDIFRX_DIR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TLO[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	THI[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **TLO[12:0]**: threshold LOW (TLO = 1.5 x UI / $T_{spdifrx\_ker\_ck}$)

This field contains the current threshold LOW estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of TLO is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:

Sampling Rate = [2 x TLO x $T_{spdifrx\_ker\_ck}$ +/- $T_{spdifrx\_ker\_ck}$] x 2/3

Note that TLO is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **THI[12:0]**: threshold HIGH (THI = 2.5 x UI / $T_{spdifrx\_ker\_ck}$)
This field contains the current threshold HIGH estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THI is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x THI x $T_{spdifrx\_ker\_ck}$ +/- $T_{spdifrx\_ker\_ck}$] x 2/5
Note that THI is updated by the hardware when SYNCD goes high, and then every frame.

54.5.10 SPDIFRX version register (SPDIFRX_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]:** SPDIFRX major revision
Major revision is 1.

Bits 3:0 **MINREV[3:0]:** SPDIFRX minor revision
Minor revision is 2.

54.5.11 SPDIFRX identification register (SPDIFRX_IPIDR)

Address offset: 0x3F8

Reset value: 0x0013 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: SPDIFRX identifier

54.5.12 SPDIFRX size identification register (SPDIFRX_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identification

These bits return the size of the memory region allocated to SPDIFRX registers.

The size of this memory region is of 1 Kbyte.

54.5.13 SPDIFRX interface register map

Table 400. SPDIFRX interface register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPDIFRX_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKSBKPEN	CKSEN	Res.	INSEL[2:0]			Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	Reset value											0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPDIFRX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEIE	SYNCDIE	SBLKIE	OVRIE	PERRIE	CSRNEIE	RXNEIE
	Reset value																										0	0	0	0	0	0	0
0x08	SPDIFRX_SR	Res.	WIDTH5[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x0C	SPDIFRX_ IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCDCF	SBDCF	OVRCF	PERRCF	Res.	Res.
	Reset value																											0	0	0	0		
0x10	SPDIFRX_ FMT0_DR	Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPDIFRX_ FMT1_DR	DR[23:0]																								Res.		PT[1:0]		C	U	V	PE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPDIFRX_ FMT2_DR	DRNL2[15:0]																DRNL1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 400. SPDIFRX interface register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	SPDIFRX_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]								USR[15:0]															
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPDIFRX_DIR	Res.	Res.	Res.	TLO[12:0]													Res.	Res.	Res.	THI[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x3F4	SPDIFRX_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	1	0
0x3F8	SPDIFRX_IPIDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x3FC	SPDIFRX_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

55 Management data input/output (MDIOS)

55.1 MDIOS introduction

An MDIO bus can be useful in systems where a master chip needs to manage (configure and get status data from) one or multiple slave chips. The bus protocol uses only two signals:

- MDC: the management data clock
- MDIO: the data line carrying the opcode (write or read), the slave (port) address, the MDIOS register address, and the data

In each transaction, the master either reads the contents of an MDIOS register in one of its slaves, or it writes data to an MDIOS register in one of its slaves.

The MDIOS peripheral serves as a slave interface to a MDIO bus. A MDIO master can use the MDC/MDIO lines to write and read 32 16-bit MDIOS registers, which are held in the MDIOS. These MDIOS registers are managed by the firmware. This allows the MDIO master to configure the application running on the STM32 and get status information from it.

The MDIOS can operate in Stop mode, optionally waking up the STM32 if the MDIO master performs a read or a write to one of its MDIOS registers.

55.2 MDIOS main features

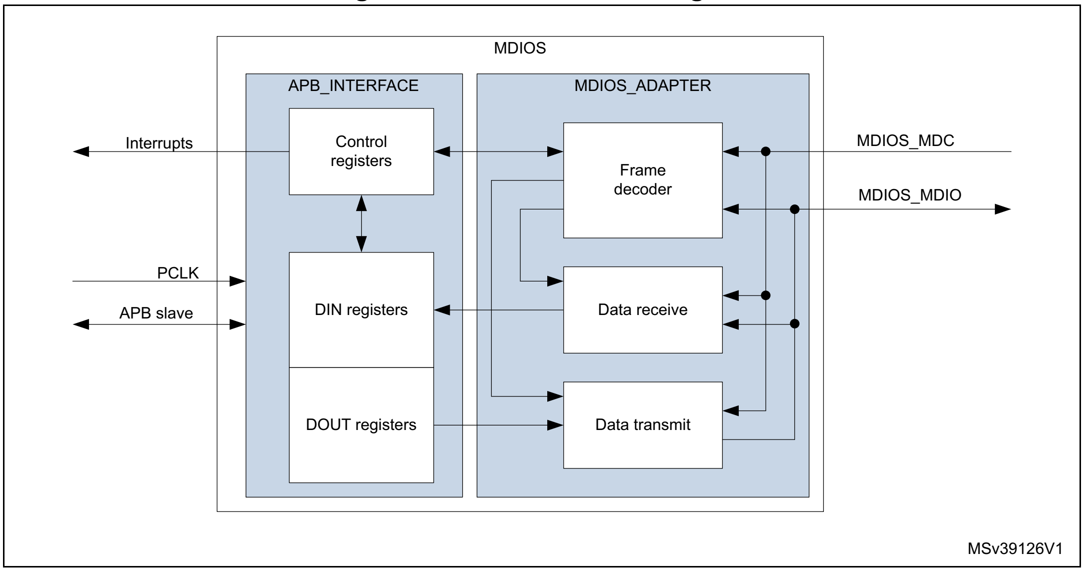
The MDIOS includes the following features:

- 32 MDIOS register addresses, each of which is managed using separate input and output data registers:
  - 32 x 16-bit firmware read/write, MDIOS read-only output data registers
  - 32 x 16-bit firmware read-only, MDIOS write-only input data registers
- Configurable slave (port) address
- Independently maskable interrupts/events:
  - MDIOS register write
  - MDIOS register read
  - MDIOS protocol error
- Able to operate in and wake up from Stop mode

55.3 MDIOS functional description

55.3.1 MDIOS block diagram

Figure 644. MDIOS block diagram



55.3.2 MDIOS protocol

The MDIOS protocol uses two signals:

- MDIOS_MDC: the clock, always driven by the master
- MDIOS_MDIO: signal carrying the opcode, address, and bidirectional data

Each transaction is performed using a frame, which contains 32 bits (each passed serially):

- 14 control bits, driven by the master
  - 2 start bits: always 01
  - 2 opcode bits: read = 10, write = 01
  - 5 port address bits, indicating which slave device is being addressed
  - 5 MDIOS register address bits, up to 32 MDIOS registers can be addressed in each slave
- 2 turnaround state bits
  - On write operations, the master drives 10
  - On read operations, the first bit is high-impedance, and the second bit is driven by the slave to 0
- 16 data bits
  - On write operations, data written to slave MDIOS register is driven by the master
  - On read operations, data read from slave MDIOS register is driven by the slave

Each frame is usually preceded by a preamble, where the MDIO stays at one for 32 MDC clocks. The master can continue to keep MDIO at one, indicating the idle condition, when it has no frame to send.

When MDIO signal is driven by the master, MDIOS samples it using the rising edge of MDC. When MDIOS drives MDIO, the output changes on the rising edge of MDC.

Figure 645. MDIO protocol write frame waveform

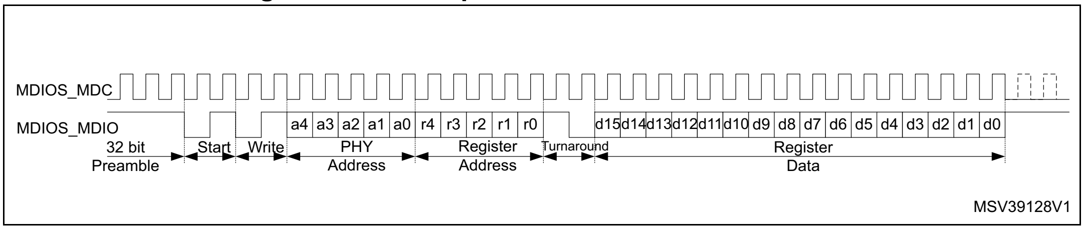
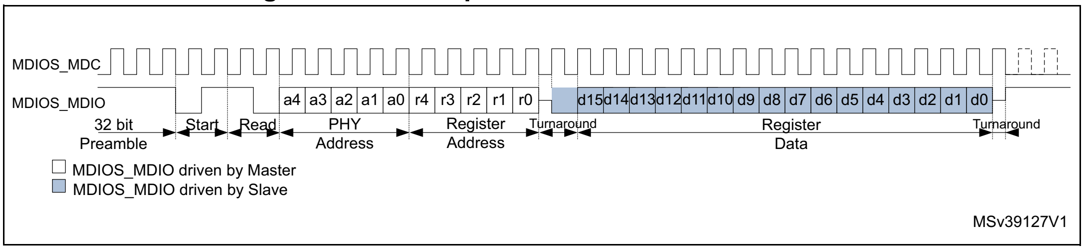


Figure 646. MDIO protocol read frame waveform



55.3.3 MDIOS enabling and disabling

The MDIOS is enabled by setting the EN bit in MDIOS_CR. When EN = 1, the MDIOS monitors the MDIO bus and service frames addressed to one of its MDIOS registers.

When the MDIOS is enabled (setting EN to one), the same write operation to MDIOS_CR must properly set the PORT_ADDRESS[4:0] field to indicate the slave port address. A frame is ignored by the MDIOS if its port address is not the same as PORT_ADDRESS[4:0] (presumably intended for another slave).

When EN = 0, the MDIOS ignores the frames being transmitted on the MDC/MDIO lines, and the IP is in a reduced consumption mode. Clearing EN also clears all of the DIN registers. If EN is cleared while the MDIOS is driving read data, it immediately releases the bus and does not drive the rest of the data. If EN is cleared while the MDIOS is receiving a frame, the frame is aborted and the data is lost.

When the MDIOS is enabled, disabled and reenabled, the status flags are not cleared. For a correct operation, the firmware must clear the status flag before reenabling the MDIOS.

55.3.4 MDIOS data

From the point of view of the MDIO master, there are 32 16-bit MDIOS registers in the MDIOS, which can be written and read. In reality, for each MDIOS register ‘x’ there are two sets of registers: MDIOS_DINRx and MDIOS_DOUTRx.

Input data

When the MDIO master transmits a frame, which writes to MDIOS register ‘x’ in the MDIOS, it is MDIOS_DINRx, which is updated with the incoming data. The registers MDIOS_DINR0 to MDIOS_DINR31 can be read by the firmware, but they can be written only by the MDIO master via the MDIO bus.

The contents of MDIOS_DINRx change immediately after the MDC rising edge when the last data bit is sampled.

If the firmware happens to read the contents of MDIOS_DINRx at the moment that it is being updated, there is a possibility that the value read is corrupted (a bit-by-bit cross between the old value and the new value). For this reason, **the firmware must assure that two subsequent reads from the same MDIOS_DINRx give the same value and assure that the data is stable when it is read**. In the very worst case, the firmware needs to read four times MDIOS_DINRx to:

1. Get the old value
2. Get an incoherent value (when reading at the moment the register changes)
3. Get the new value
4. Confirm the new value

If the firmware uses the WRF interrupt and can guarantee that it reads the MDIOS_DINRx register before any new MDIOS write frame completes, the firmware can perform a single read.

If the MDIO master performs a write operation with a register address, which is greater than 31, the MDIOS ignores the frame (the data is not saved and no flag is set).

Output data

When the MDIOS receives a frame, which requests to read register ‘x’, it returns the value found in MDIOS_DOUTRx. Thus, if the MDIO master expects to read the same value, which it previously wrote to MDIOS register ‘x’, the firmware must copy the data from MDIOS_DINRx to MDIOS_DOUTRx each time new data is written to MDIOS_DINRx. For correct operation, the firmware must copy the data to the MDIOS_DOUTRx register within a preamble (if the master sends preambles before each frame) plus 15 cycles time.

When an MDIOS register is read via the MDIO bus, the MDIOS passes the 16-bit value (from the corresponding MDIOS_DOUTRx) to the MDIOS clock domain during the 15th cycle of the read frame. If the firmware attempts to write MDIOS_DOUTRx while the MDIO master is currently reading MDIOS register ‘x’, the firmware write operation is ignored if it occurs during the 15th cycle of the frame (during a one-MDC-cycle window). Therefore, **after writing MDIOS_DOUTRx, the firmware must read back the same MDIOS_DOUTRx and confirm that the value was actually written**. If MDIOS_DOUTRx does not contain the value, which was written, the firmware can simply try writing and rereading again.

If the MDIOS frequency is very slow compared to the mdios_pclk frequency, it is better not to tie up the CPU by continuously writing and rereading MDIOS_DOUTRx.

Note: The read flag (RDFx) is set as soon as the MDIOS_DOUTRx DOUT[15:0] value is passed to the MDIOS clock domain. Thus, when a write to MDIOS_DOUTRx is ignored (when the value read back is not the value, which was just written), the firmware can use a read interrupt to know when it is able to write MDIOS_DOUTRx.

Respect the following procedure if the MDC clock is very slow:

1. Write MDIOS_DOUTRx.
2. Assure that all of the read flags are zero (MDIOS_RDFR = 0x0000). Clear the flags if necessary using MDIOS_CRDFR.
3. Read back the same MDIOS_DOUTRx and compare the value with the value, which was written in step 1.

4. If the values are the same, the procedure is done. Otherwise, continue to step 5.
5. Enable read interrupts by setting the RDIE bit in MDIOS_CR1.
6. In the interrupt routine, assure that RDFx is set (no other read flags are set before bit x).
7. There is a 31 cycle + preamble time window (if the master sends a preamble before each frame) to write MDIOS_DOUTRx safely without doing a read-back and compare. If this maximum delay cannot be guaranteed, go back to step 1.

If the MDIO master performs a read operation with a register address, which is greater than 31, the MDIOS returns a data value of all zeros.

55.3.5 MDIOS APB frequency

Whenever the firmware reads from MDIOS_DINRx or writes to MDIOS_DOUTRx, the frequency of the APB bus must be at least 1.5 times the MDC frequency. For example, if MDC is at 20 MHz, the APB must be at 30 MHz or faster.

55.3.6 Write/read flags and interrupts

When MDIOS register ‘x’ is written via the MDIO bus, the WRFx bit in MDIOS_WRFR is set. WRFx becomes one at the moment that MDIOS_DINRx is updated, which is when the last data bit is sampled on a write frame. An interrupt is generated if WRIEN = 1 (in MDIOS_CR). WRFx is cleared by software by writing 1 to CWRFx (in MDIOS_CWRFR).

When MDIOS register ‘x’ is read via the MDIO bus, the RDFx bit in MDIOS_RDFR is set. RDFx becomes one at the moment that MDIOS_DOUTRx is copied to the MDC clock domain, which is on the 15th cycle of a read frame. An interrupt is generated if RDIEN = 1 (in MDIOS_CR). RDFx is cleared by software by writing 1 to CRDFx (in MDIOS_CRDFR).

55.3.7 MDIOS error management

There are three types of errors with their corresponding error flags:

- Preamble error: PERF (bit 0 of MDIOS_SR)
- Start error: SERF (bit 1 of MDIOS_SR)
- Turnaround error: TERF (bit 2 of MDIOS_SR)

Each error flag is set by hardware when the corresponding error condition occurs. Each flag can be cleared by writing ‘1’ to the corresponding bit in the clear flag register (MDIOS_CLRFR).

An interrupt occurs if any of the three error flags is set while EIE = 1 (MDIOS_CR).

Besides, setting an error flag, the MDIOS performs no action for a frame in which an error is detected: MDIOS_DINRx are not updated and the MDIO line is not forced during the data phase.

For a given frame, errors do not accumulate. For example, if a preamble error is detected, no check is done for a start error or a turnaround error for the rest of the current frame.

When DPC = 0, following a detected error, all new frames, and errors are ignored until a complete full preamble has been detected.

When DPC = 1 (disable preamble check, MDIOS_CR[7]), all frames and new errors are ignored as long as one of the error flags is set. As soon as the error bit is cleared, the MDIOS starts looking for a start sequence. Thus, the application must clear the error flag only when it is sure that no frame is currently in progress. Otherwise, the MDIOS likely misinterprets the bits being sent and becomes desynchronized with the master.

Preamble errors

A preamble error occurs when a start sequence begins (with MDIO sampled at 0) without being immediately preceded by a preamble (MDIO sampled at 1 for at least 32 consecutive clocks).

Preamble errors are not reported after the MDIOS is first enabled (EN = 1 in MDIOS_CR) until after a full preamble is received. This is to avoid an error condition when the peripheral frame detection is enabled while a preamble or frame is already in progress. In this case, the MDIOS ignores the first frame (since it did not first detect a full preamble), but does not set PERF.

If the DPC bit (disable preamble check, MDIOS_CR[7]) is set, the MDIO master can send frames without preceding preambles and no preamble error is signaled. When DPC = 1, the application must assure that the master is not in the process of sending a frame at the moment that the MDIOS is enabled (EN is set). Otherwise, the slave might become desynchronized with the master.

Start errors

A start error occurs when an illegal start sequence occurs or if an illegal command is given. The start sequence must always be 01, and the command must be either 01 (write) or 10 (read).

As with preamble errors, start errors are not reported until after a full preamble is received.

Turnaround errors

A turnaround error occurs when an error is detected in the turnaround bits of write frames. The 15^{th} bit of the write frame must be 1 and the 16^{th} bit must be 0.

Turnaround errors are only reported after a full preamble is received, there is no start error, the port address in the current frame matches, and the register address is in the supported range 0 to 31.

55.3.8 MDIOS in Stop mode

The MDIOS can operate in Stop mode, responding to all reads, performing all writes, and causing the STM32 to wake up from Stop mode on MDIOS interrupts.

55.3.9 MDIOS interrupts

There is a single interrupt vector for the three types of interrupts (write, read, and error). Any of these interrupt sources can wake the STM32 up from Stop mode. All interrupt flags need to be cleared in order to clear the interrupt line.

Table 401. Interrupt control bits

Interrupt event	Event flag	Enable control bit
Write interrupt	WRF[31:0]	WRIE
Read interrupt	RDF[31:0]	RDIE
Error interrupt	PERF (preamble), SERF (start), TERF (turnaround)	EIE

55.4 MDIOS registers

55.4.1 MDIOS configuration register (MDIOS_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PORT_ADDRESS[4:0]					DPC	Res.	Res.	Res.	EIE	RDIE	WRIE	EN
			rw	rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **PORT_ADDRESS[4:0]**: slave address

Can be written only when the peripheral is disabled (EN = 0).
If the address given by the MDIO master matches PORT_ADRESS[4:0], the MDIOS services the frame. Otherwise, the frame is ignored.

Bit 7 **DPC**: disable preamble check

0: MDIO master must give preamble before each frame.
1: MDIO master can send each frame without a preceding preamble, and the MDIOS does not signal a preamble error.

Note: When this bit is set, the application must be sure that no frame is currently in progress when the MDIOS is enabled. Otherwise, the MDIOS can become desynchronized with the master.

This bit cannot be changed unless EN = 0 (though it can be changed at the same time that EN is being set).

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **EIE**: error interrupt enable

0: Interrupt is disabled.
1: Interrupt is enabled.

Note: When this bit is set, an interrupt is generated if any of the error flags (PERF, SERF, or TERF in the MDIOS_SR register) is set.

Bit 2 **RDIE**: register read interrupt enable

0: Interrupt is disabled.
1: Interrupt is enabled.

Note: When this bit is set, an interrupt is generated if any of the read flags (RDF[31:0] in the MDIOS_RDFR register) is set.

Bit 1 **WRIE**: register write interrupt enable

0: Interrupt is disabled.
1: Interrupt is enabled.

Note: When this bit is set, an interrupt is generated if any of the read flags (WRF[31:0] in the MDIOS_WRFR register) is set.

Bit 0 **EN**: peripheral enable

0: MDIOS is disabled.
1: MDIOS is enabled and monitoring the MDIO bus (MDC/MDIO).

55.4.2 MDIOS write flag register (MDIOS_WRFR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WRF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRF[31:0]**: write flags for MDIOS registers 0 to 31.

Each bit is set by hardware when the MDIO master performs a write to the corresponding MDIOS register. An interrupt is generated if WRIE (in MDIOS_CR) is set.

Each bit is cleared by software by writing ‘1’ to the corresponding CWRF bit in the MDIOS_CWRFR register.

For WRFx:

0: MDIOS register x is not written by the MDIO master.

1: MDIOS register x is written by the MDIO master and the data is available in DIN[15:0] in the MDIOS_DINRx register.

55.4.3 MDIOS clear write flag register (MDIOS_CWRFR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CWRF[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CWRF[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **CWRF[31:0]**: clear the write flag

Writing 1 to CWRFx clears the WRFx bit in the MDIOS_WRF register.

55.4.4 MDIOS read flag register (MDIOS_RDFR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDF[31:0]**: read flags for MDIOS registers 0 to 31.
Each bit is set by hardware when the MDIO master performs a read from the corresponding MDIOS register. An interrupt is generated if RDIE (in MDIOS_CR) is set.
Each bit is cleared by software by writing one to the corresponding CRDF bit in the MDIOS_CRDFR register.
For RDFx:
0: MDIOS register x is not read by the MDIO master.
1: MDIOS register x is read by the MDIO master.

55.4.5 MDIOS clear read flag register (MDIOS_CRDFR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRDF[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRDF[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **CRDF[31:0]**: clear the read flag
Writing 1 to CRDFx clears the RDFx bit in the MDIOS_RDF register.

55.4.6 MDIOS status register (MDIOS_SR)

Address offset: 0x014

Reset value: 0x0000 0000

Writes to this register have no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERF	SERF	PERF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TERF**: turnaround error flag
0: No turnaround error has occurred.
1: A turnaround error has occurred.
Note: Writing 1 to CTERF (MDIOS_CLRFR) clears this bit.

Bit 1 **SERF**: start error flag
0: No start error has occurred.
1: A start error has occurred.
Note: Writing 1 to CSERF (MDIOS_CLRFR) clears this bit.

Bit 0 **PERF**: preamble error flag
0: No preamble error has occurred.
1: A preamble error has occurred.
Note: Writing 1 to CPERF (MDIOS_CLRFR) clears this bit.
This bit is not set if DPC (disable preamble check, MDIOS_CR[7]) is set.

55.4.7 MDIOS clear flag register (MDIOS_CLRFR)

Address offset: 0x018

Reset value: 0x0000 0000

Reads on this register returns all zeros.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTERF	CSERF	CPERF
													rc_w1	rc_w1	rc_w1

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CTERF**: clear the turnaround error flag
Writing 1 to this bit clears the TERF flag (in MDIOS_SR).
When DPC = 1 (MDIOS_CR[7]), the TERF flag must be cleared only when there is not a frame already in progress.

Bit 1 **CSERF**: clear the start error flag
Writing 1 to this bit clears the SERF flag (in MDIOS_SR).
When DPC = 1 (MDIOS_CR[7]), the SERF flag must be cleared only when there is not a frame already in progress.

Bit 0 **CPERF**: clear the preamble error flag
Writing 1 to this bit clears the PERF flag (in MDIOS_SR).

55.4.8 MDIOS input data register x (MDIOS_DINRx)

Address offset: 0x100 + 0x4 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIN[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DIN[15:0]**: input data received from MDIO master during write frames
This field is written by hardware with the 16-bit data received in a write frame, which is addressed to MDIOS register x.

55.4.9 MDIOS output data register x (MDIOS_DOUTRx)

Address offset: 0x180 + 0x4 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DOUT[15:0]**: output data sent to MDIO master during read frames
This field is written by software. These 16 bits are serially output on the MDIO bus during read frames, which address the MDIOS register x.

55.4.10 MDIOS hardware configuration register (MDIOS_HWCFGR)

Address offset: 0x03F0

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBREG[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **NBREG[7:0]**: MDIOS configuration number of registers (N).

55.4.11 MDIOS version register (MDIOS_VERR)

Address offset: 0x03F4

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: MDIOS major revision
This field returns major revision of the MDIOS.

Bits 3:0 **MINREV[3:0]**: minor revision
This field returns the minor revision of the MDIOS.

55.4.12 MDIOS identification register (MDIOS_IPIDR)

Address offset: 0x03F8

Reset value: 0x0018 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: identification code
This field returns the identification code of the MDIOS.

55.4.13 MDIOS size identification register (MDIOS_SIDR)

Address offset: 0x03FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size ID
This field returns the size ID of the MDIOS.

55.4.14 MDIOS register map

Table 402. MDIOS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	MDIOS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_ADDRESS[4:0]					DPC	Res.	Res.	Res.	EIE	RDIE	WRIE	EN
	Reset value																				0	0	0	0	0	0				0	0	0	0
0x004	MDIOS_WRFR	WRF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 402. MDIOS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	MDIOS_CWRFR	CWRF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	MDIOS_RDFR	RDF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	MDIOS_CRDFR	CRDF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	MDIOS_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERF	SERF	PERF
	Reset value																														0	0	0
0x018	MDIOS_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTERF	CSERF	CPERF
	Reset value																														0	0	0
0x01C-0x0FC	Reserved	Reserved																															
0x100 + 0x04 * x (x=0 to 31)	MDIOS_DINRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIN[15:0]															
Last address: 0x17C	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x180 + 0x4 * x (x=0 to 31)	MDIOS_DOUTRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOUT[15:0]															
Last address: 0x1FC	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x200 - 0x3EC	Reserved	Reserved																															
0x3F0	MDIOS_HWCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBREG[7:0]							
	Reset value																									0	0	1	0	0	0	0	0
0x3F4	MDIOS_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0x3F8	MDIOS_IDR	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x3FC	MDIOS_SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5* for the register boundary addresses.

56 Secure digital input/output MultiMediaCard interface (SDMMC)

56.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*.
  Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit.
  (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0.*
  (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.*
  Card support for two different databus modes: 1-bit (default) and 4-bit.
  (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode.
  (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

56.2 SDMMC implementation

Table 403. SDMMC features

SDMMC modes/features[1]	SDMMC1	SDMMC2	SDMMC3
Variable delay	X	X	X
SDMMC_CKIN	X	X	X
SDMMC_CDIR, SDMMC_D0DIR	X	X	X
SDMMC_D123DIR	X	X	X

1. X = supported.

56.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 647. SDMMC “no response” and “no data” operations

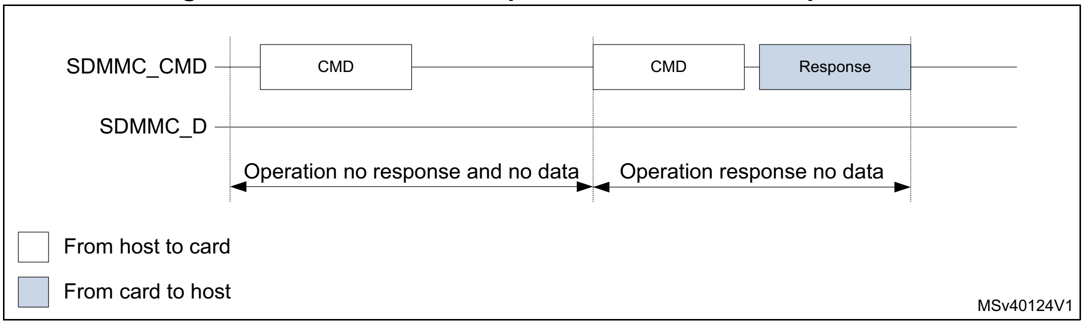
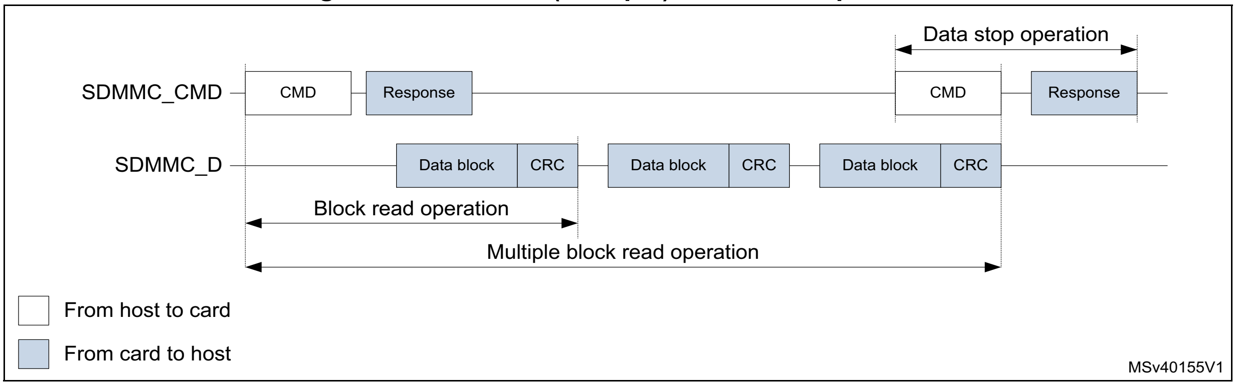
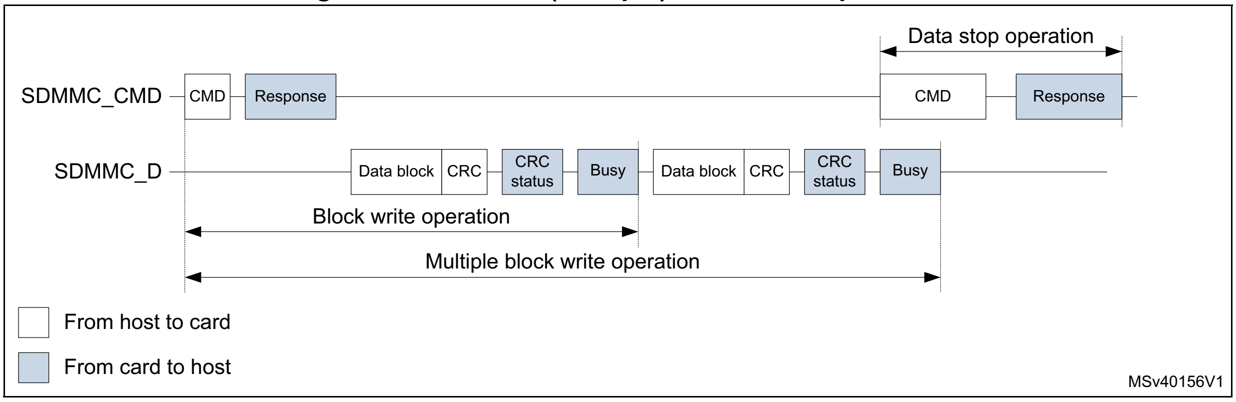


Figure 648. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a e•MMC multiple block read with predefined block count.

Figure 649. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 650. SDMMC (sequential) stream read operation

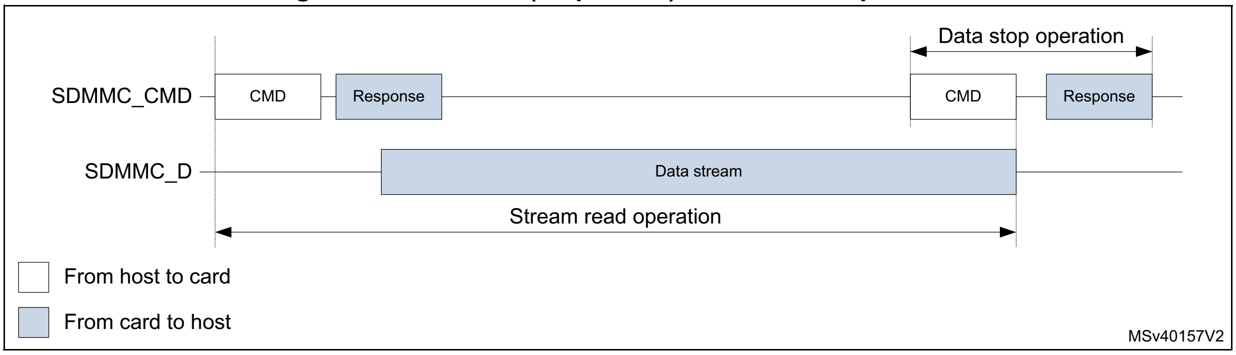
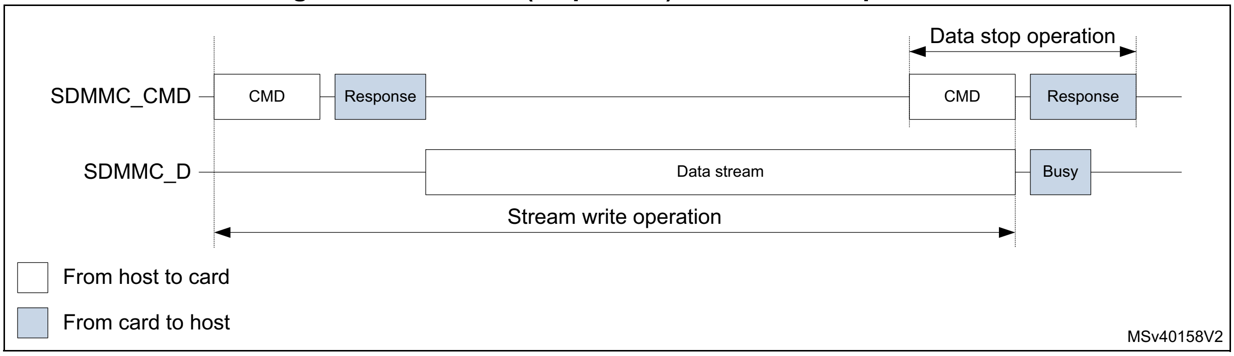


Figure 651. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

56.4 SDMMC operation modes

Table 404. SDMMC operation modes SD and SDIO

SDIO bus speed modes(1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 405. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V](5)
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.

56.5 SDMMC functional description

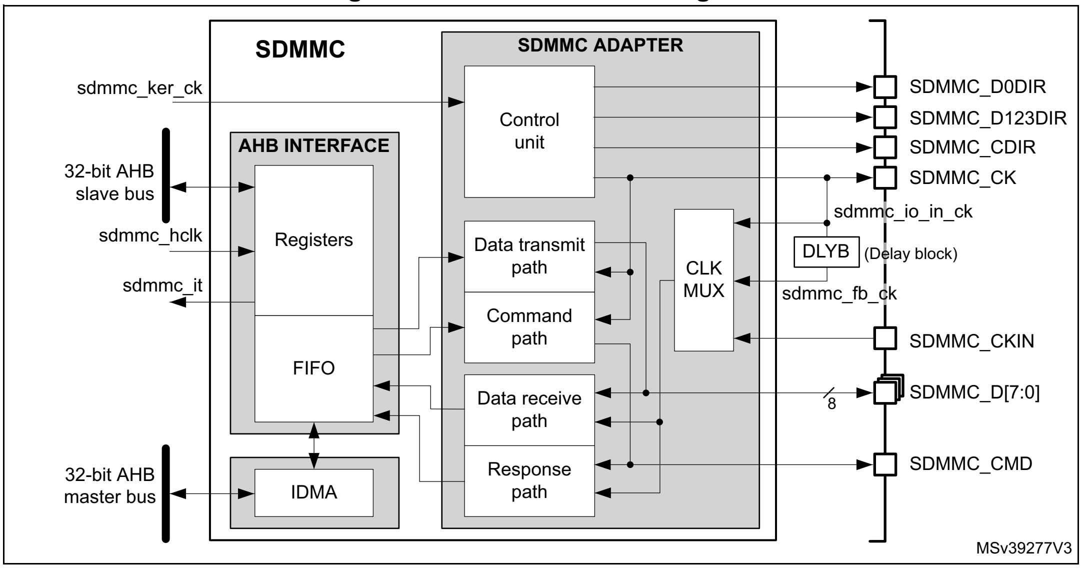
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the e•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

56.5.1 SDMMC block diagram

Figure 652 shows the SDMMC block diagram.

Figure 652. SDMMC block diagram



56.5.2 SDMMC pins and internal signals

Table 406 lists the SDMMC internal input/output signals, *Table 407* the SDMMC pins (alternate functions).

Table 406. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt

Table 406. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 407. SDMMC pins

Pin name	Pin type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

56.5.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 653*.

Figure 653. SDMMC Command and data phase relation

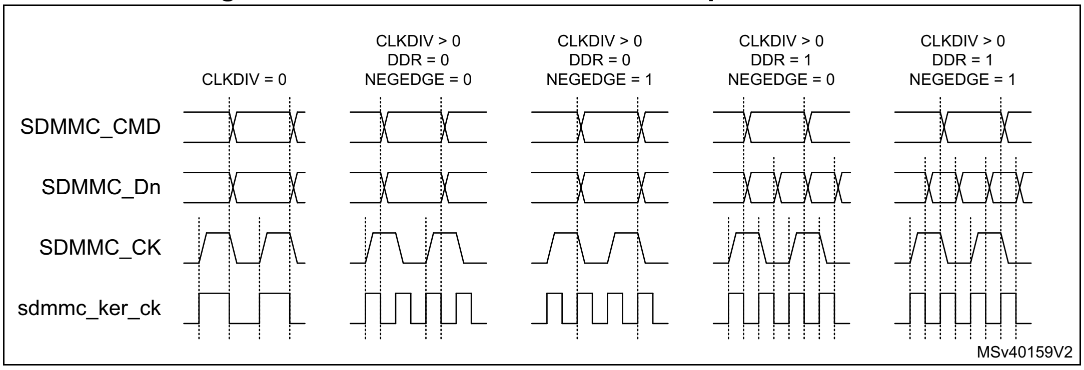


Table 408. SDMMC Command and data phase selection

<table>
<tr><th>CLKDIV</th><th>DDR</th><th>NEGEDGE</th><th>SDMMC_CK</th><th>Command out</th><th>Data out</th></tr>
<tr><td>0</td><td>x</td><td>x</td><td>= sdmmc_ker_ck</td><td colspan="2">Generated on sdmmc_ker_ck falling edge</td></tr>
<tr><td rowspan="4">>0</td><td rowspan="2">0</td><td>0</td><td rowspan="4">Generated on sdmmc_ker_ck rising edge</td><td colspan="2">Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.</td></tr>
<tr><td>1</td><td colspan="2">Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.</td><td rowspan="2">Generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.</td></tr>
<tr><td>1</td><td>Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.</td></tr>
</table>

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

56.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 652: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

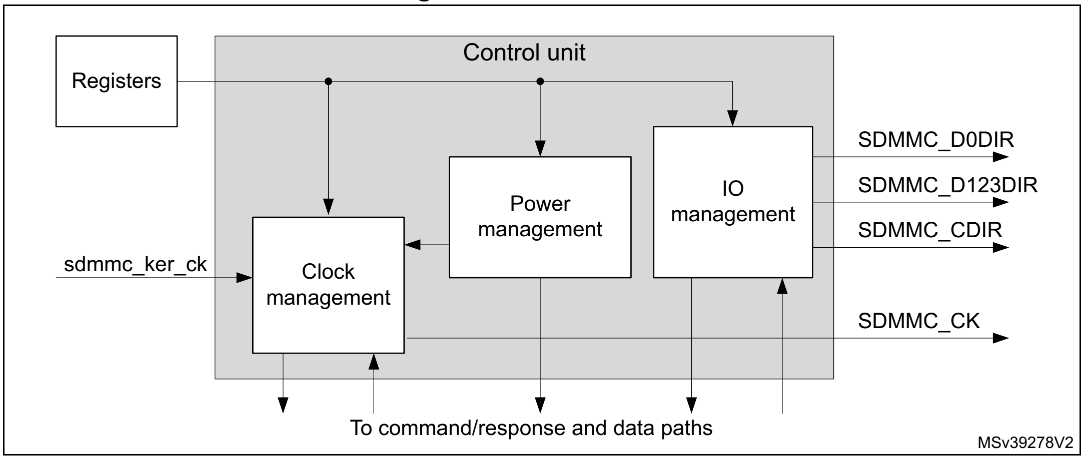
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 654*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 654. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:

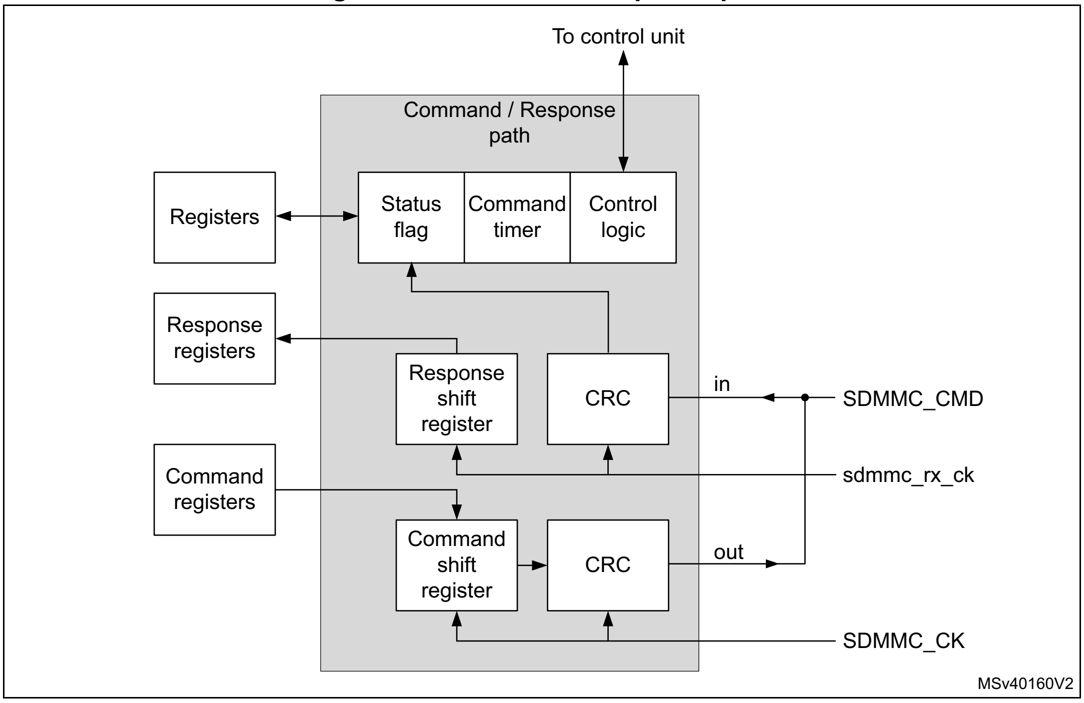
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 655. Command/response path

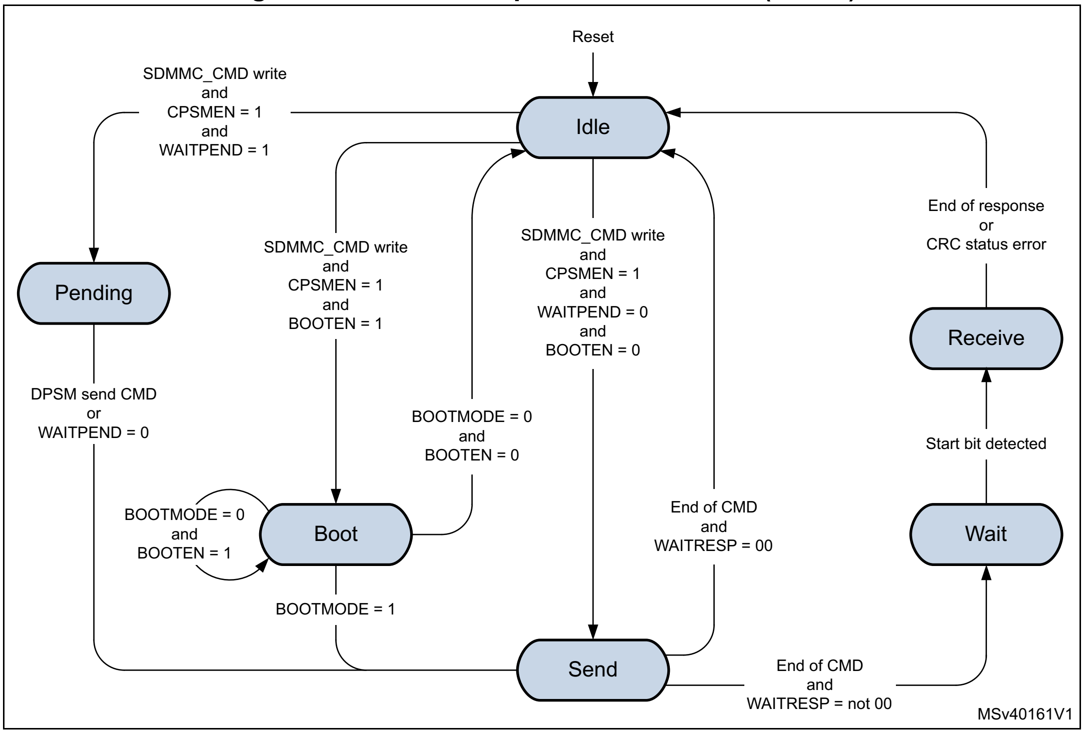


Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.

Figure 656. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

  – When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 409.*

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

Table 409. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 410*, *Table 411* and *Table 412*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 410. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 411. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 412. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 653*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

$CRC[6:0] = \text{remainder} [(M(x) * x^7) / G(x)]$

$G(x) = x^7 + x^3 + 1$

$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit before CRC}) * x^0$

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 413*.

Table 413. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start voltage switch sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 414*:

Table 414. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response (CPSM moves from Send to Idle).
CMDREND	Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).
CCRCFAIL	Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).
CTIMEOUT	Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).
CPSMACT	Command transfer in progress (CPSM not in Idle state).

The command path error handling is shown in *Table 415*:

Table 415. Command path error handling

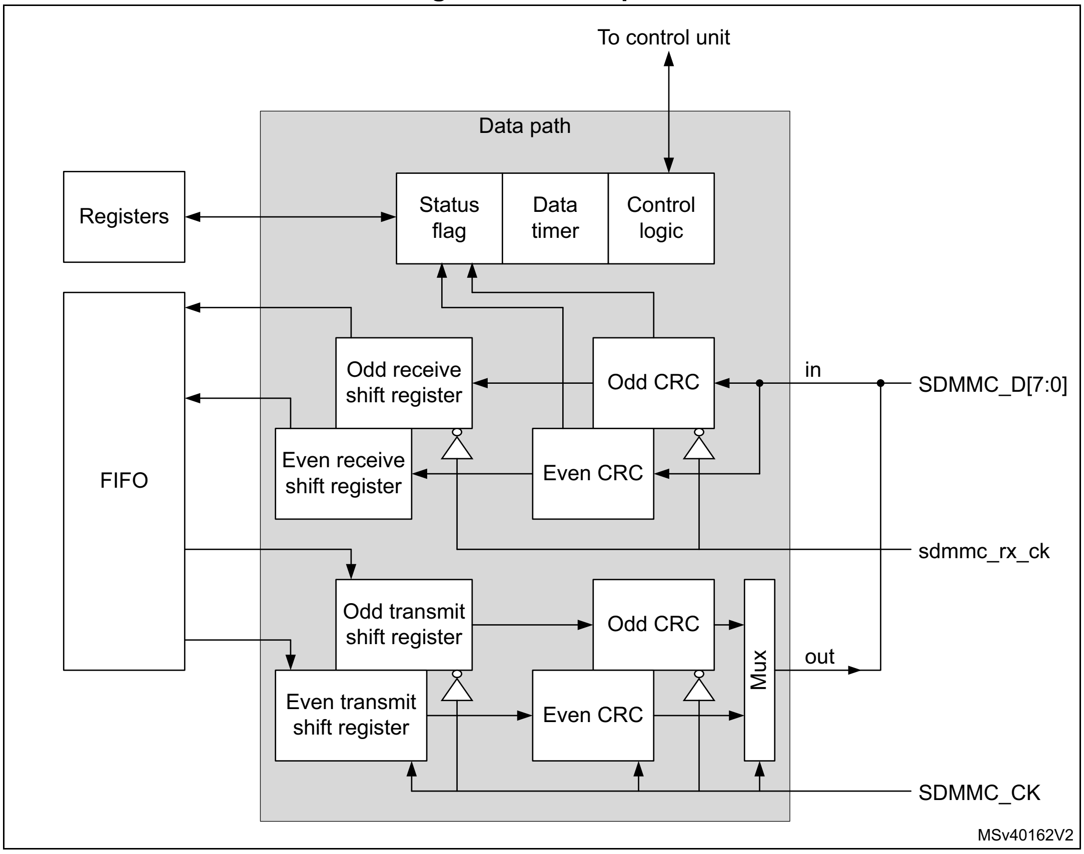
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 657* shows the data path block diagram.

Figure 657. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 658* and *Figure 659*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

Figure 658. DDR mode data packet clocking

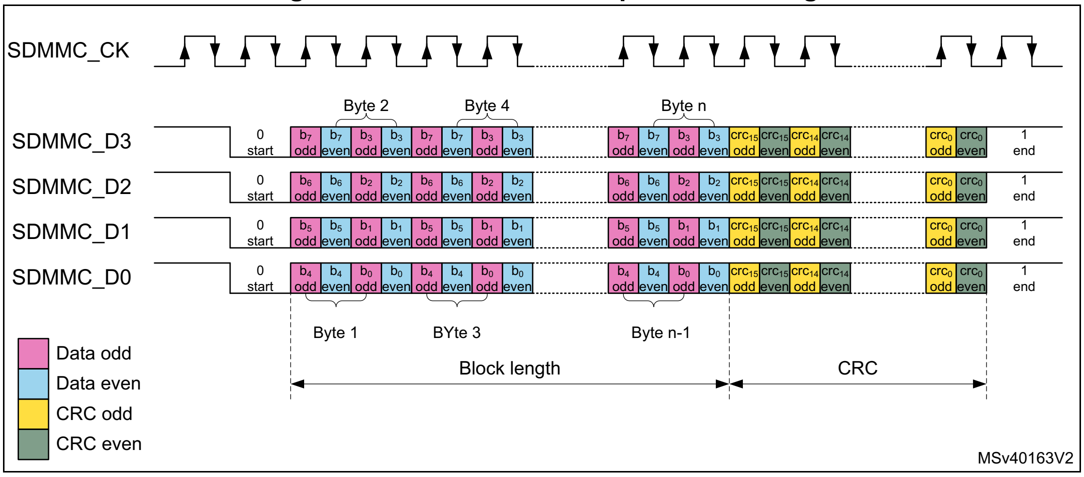
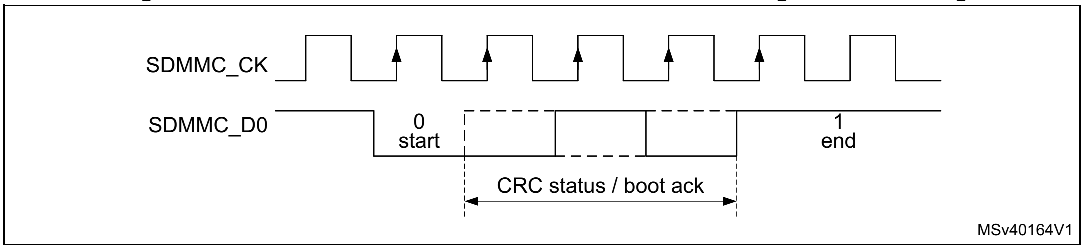


Figure 659. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

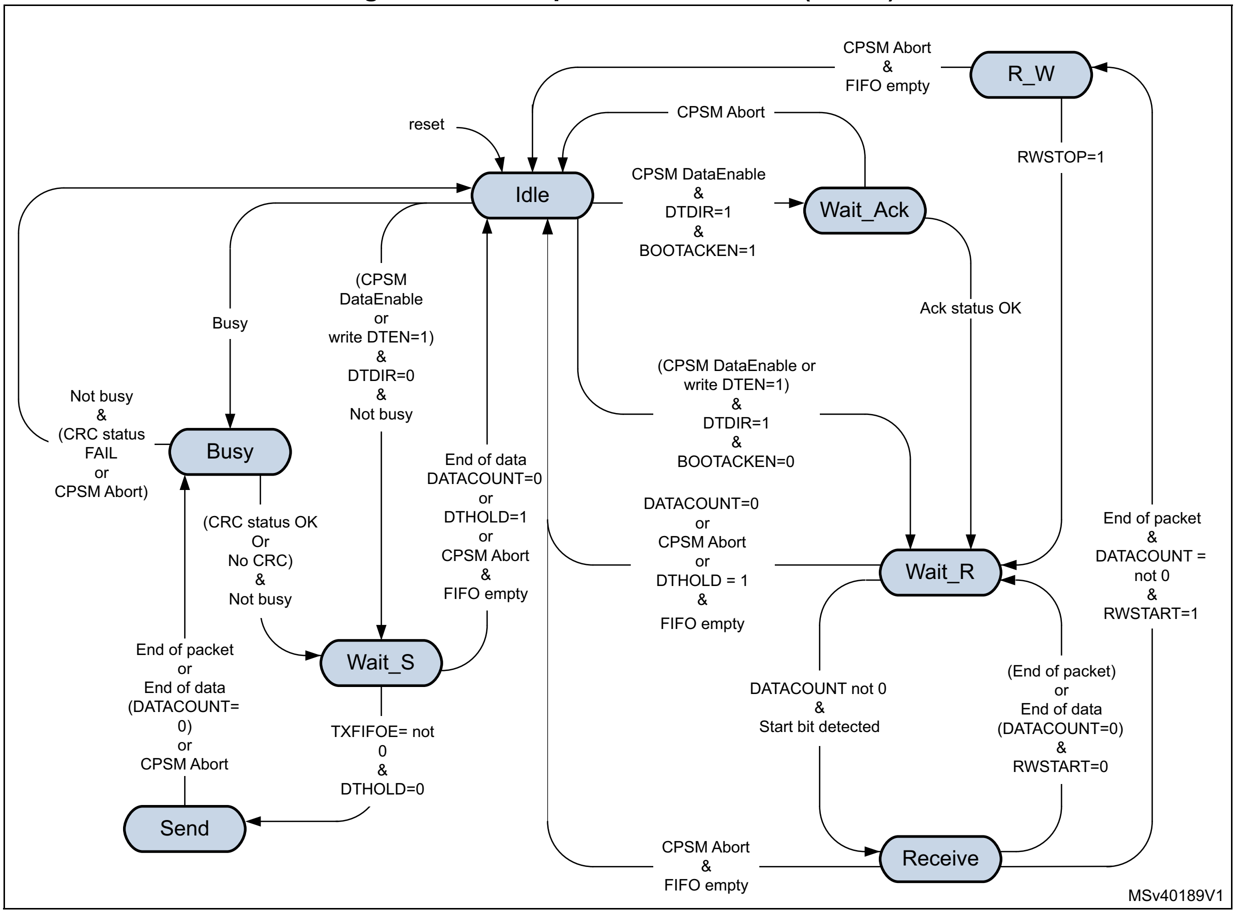
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 660*. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

Figure 660. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:
  - When the data direction bit (DTDIR) indicates send, moves to the Wait_S.
  - When the data direction bit (DTDIR) indicates receive, moves to the
    - Wait_R when BOOTACKEN register bit is clear.
    - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.

- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.
 - When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.
- **R_W** state: the data path Read Wait the bus.
  - The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  - If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.
  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise.
  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  - When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  - When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0:

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 416. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	Yes	1	00
SDIO multibyte	0	(DATALENGTH)	Yes	1	01
e•MMC stream	0	(DATALENGTH)	No	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 417*:

Table 417. Data path status flags and clear bits

<table>
<tr><th>Flag</th><th></th><th>Description</th></tr>
<tr><td rowspan="3">DATAEND</td><td>TX</td><td>Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)</td></tr>
<tr><td>RX</td><td rowspan="2">Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)</td></tr>
<tr><td>Boot</td></tr>
</table>

Table 417. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX Boot	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX Boot	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX RX Boot	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 418*:

Table 418. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 419*.

Table 419. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
3. Enable the data transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   – The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.

- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 420*.

Table 420. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 421*.

Table 421. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

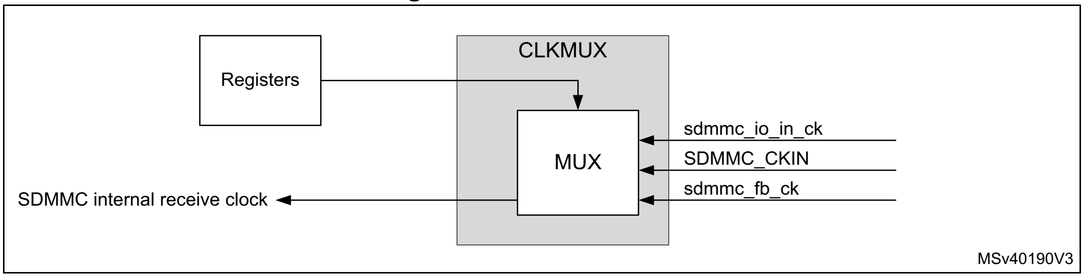
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 661. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

56.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.
- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

56.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer.

When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

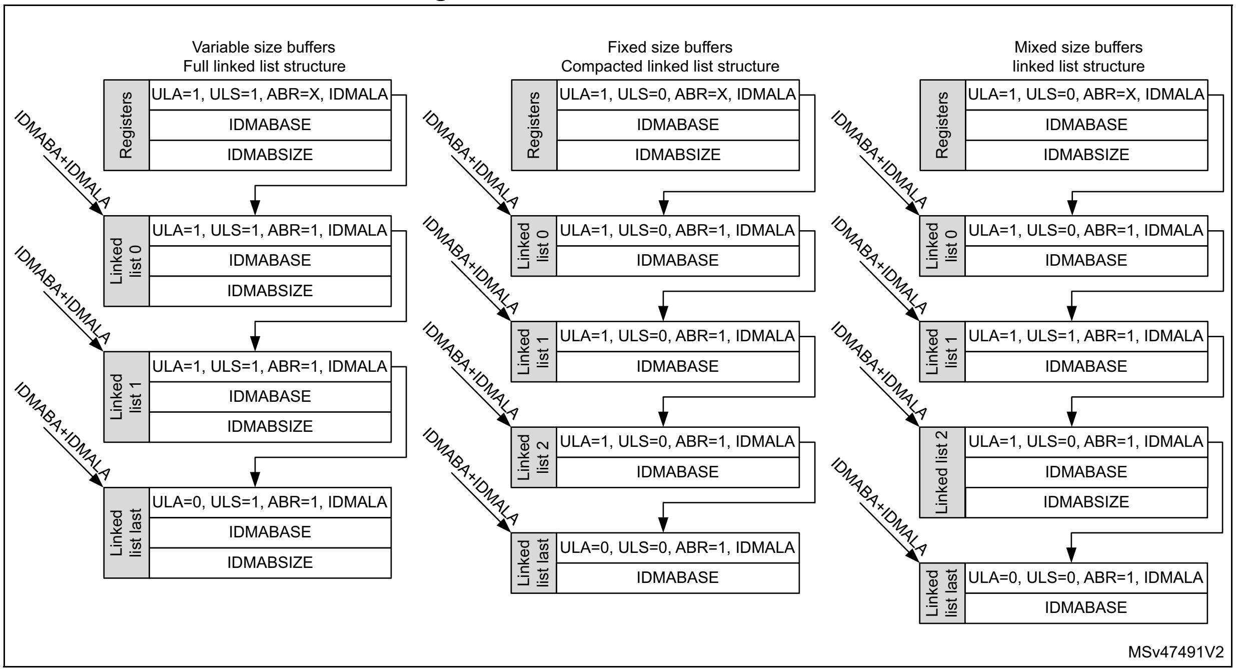
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 662. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

56.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth: for example, for SDR50 4-bit mode (50 Mbyte/s), the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

Table 422. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	19.5
e•MMC HS	8	52	39
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	150
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5
SD SDR104	4	208	78

56.6 Card functional description

56.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 423. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

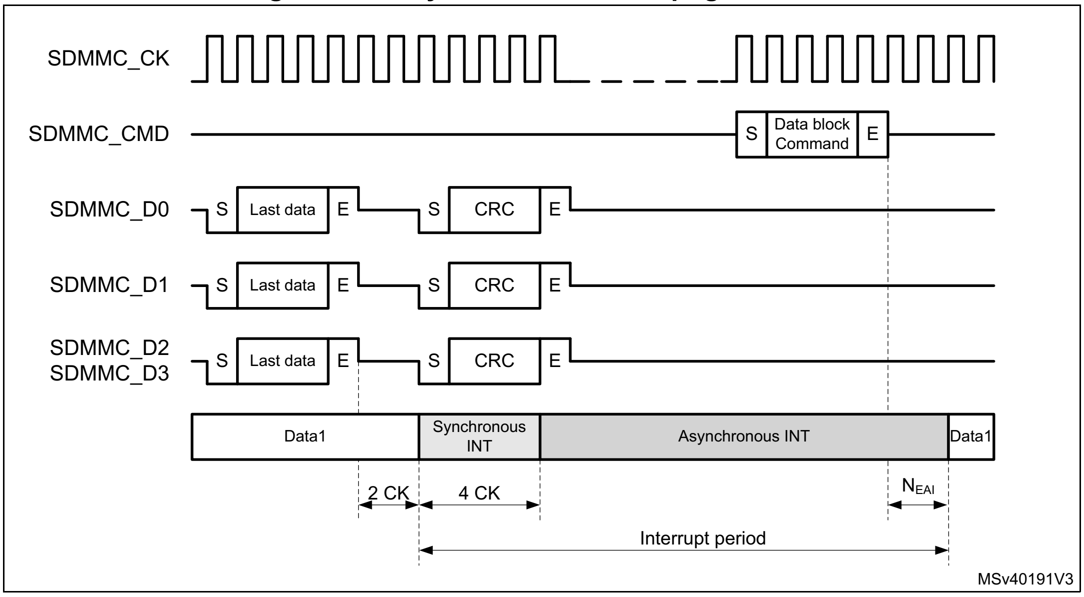
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 423*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 663. Asynchronous interrupt generation



The timing of the interrupt period is depending on the bus speed mode.

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 664. Synchronous interrupt period data read

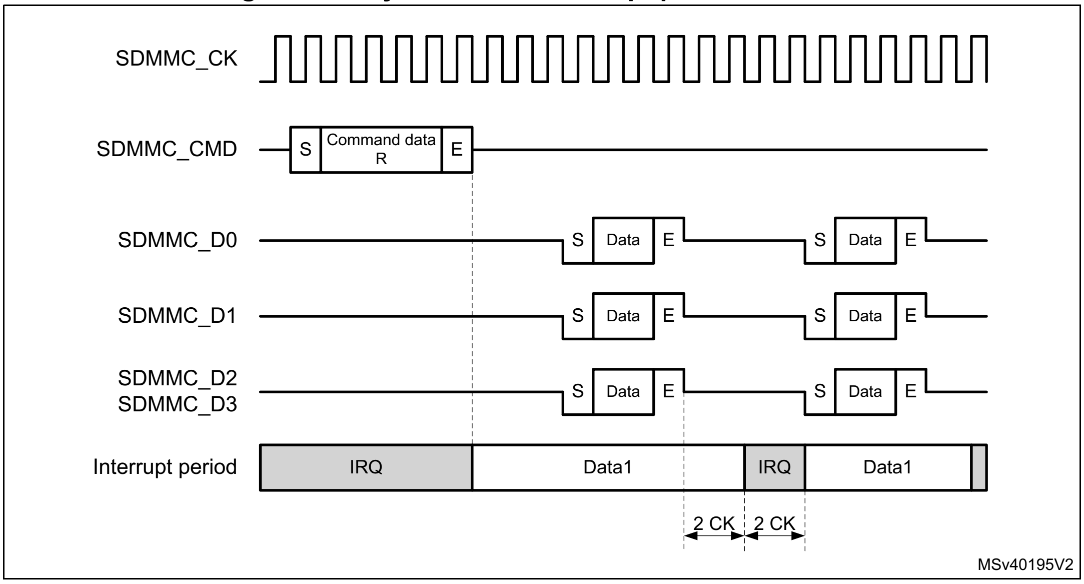
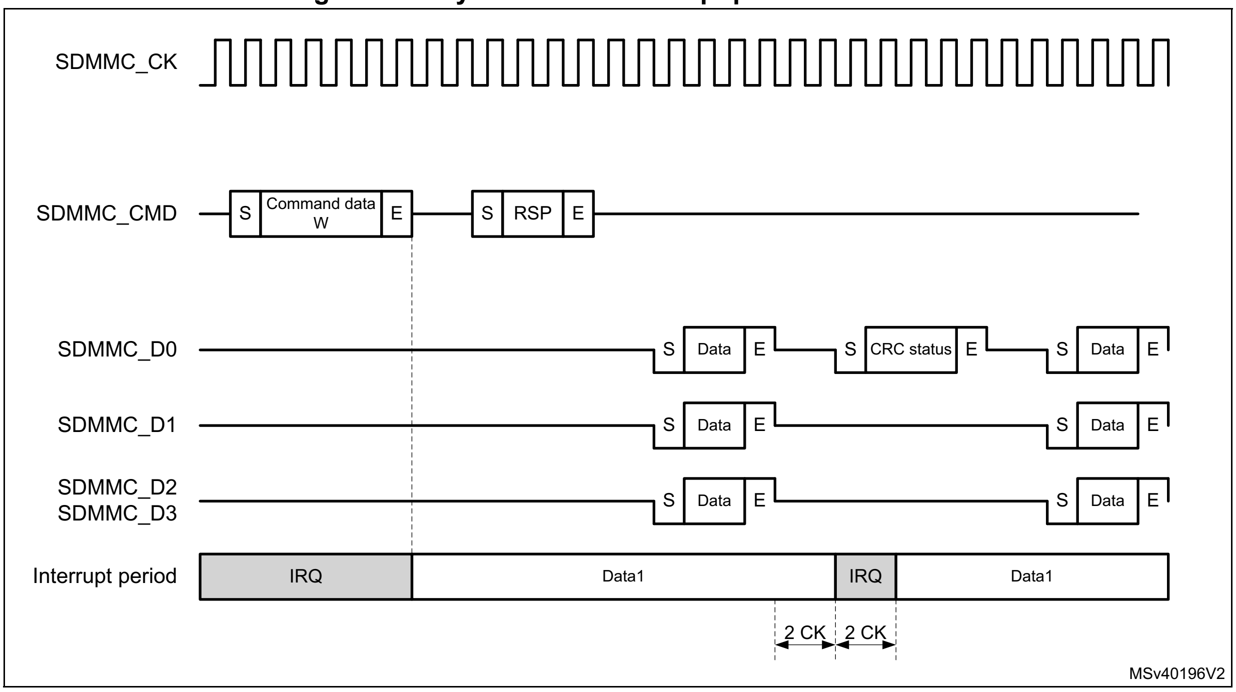


Figure 665. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 666. Asynchronous interrupt period data read

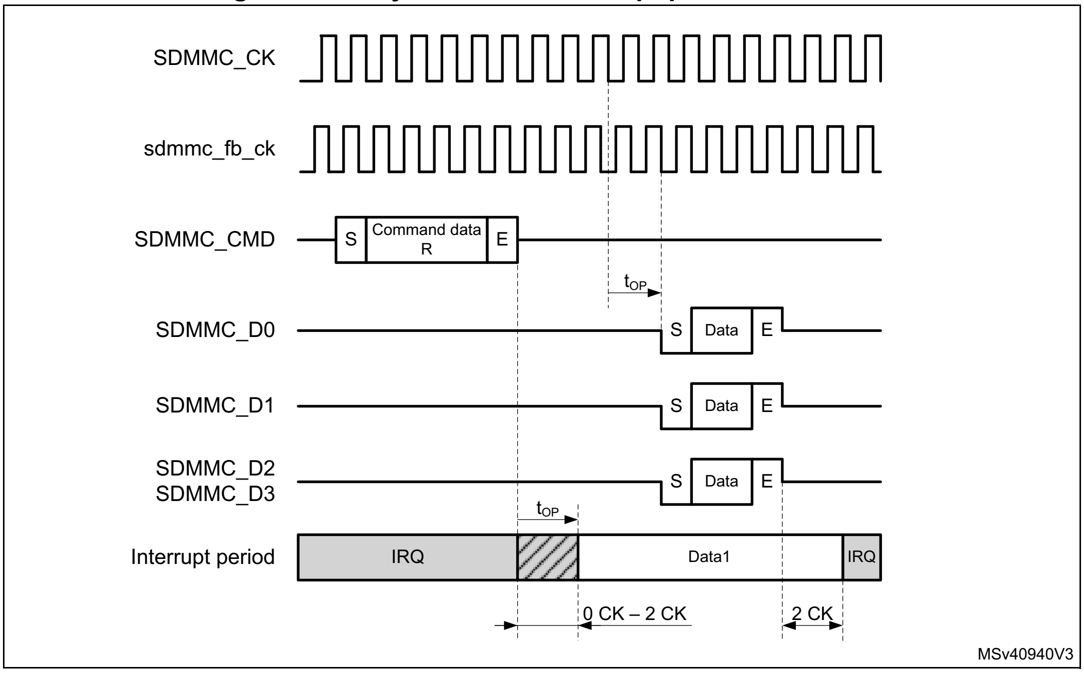
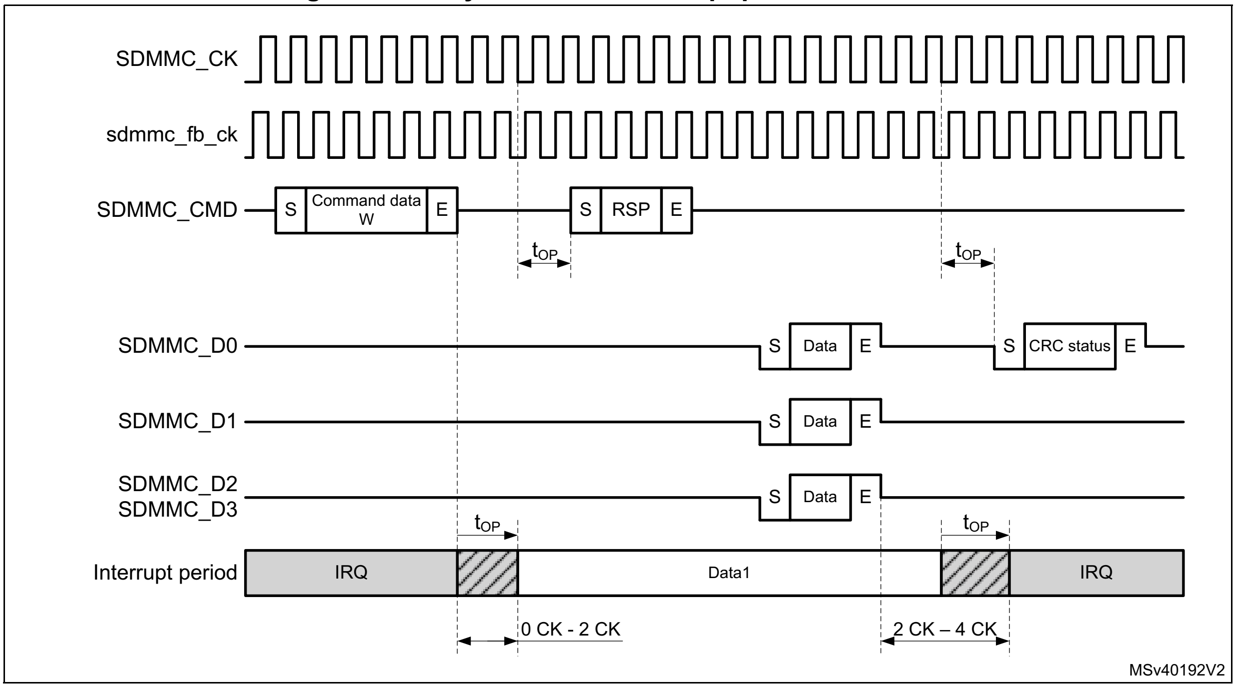


Figure 667. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE "block data transfer ending with STOP_TRANSMISSION command", the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 424*.

Table 424. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are two methods to pause the data transfer during the block gap:

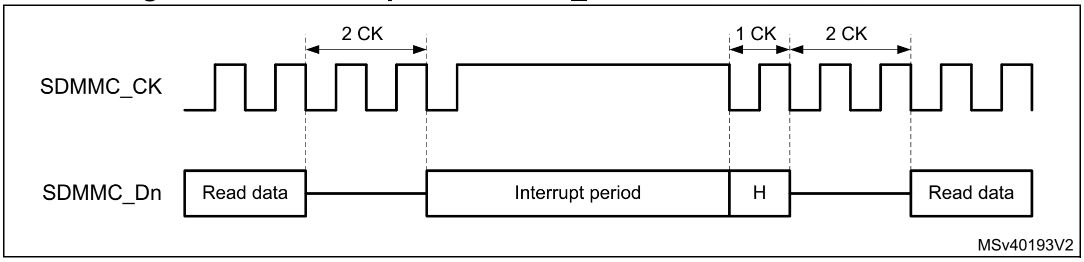
1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according *Table 423*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

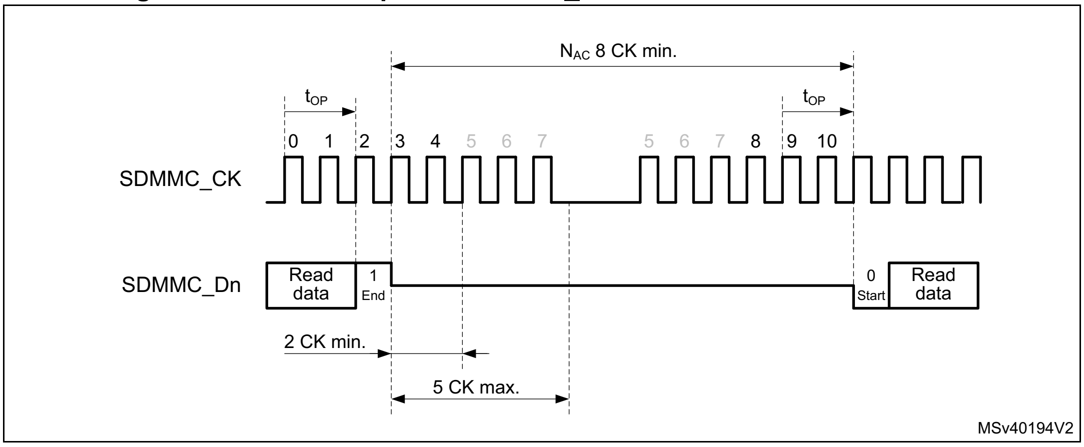
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see *Figure 668*).

Figure 668. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 669*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 669. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



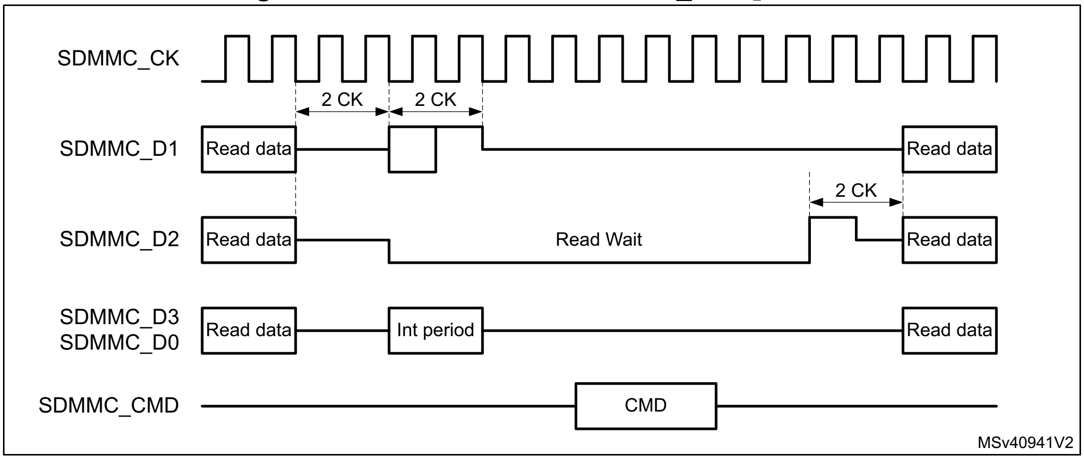
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

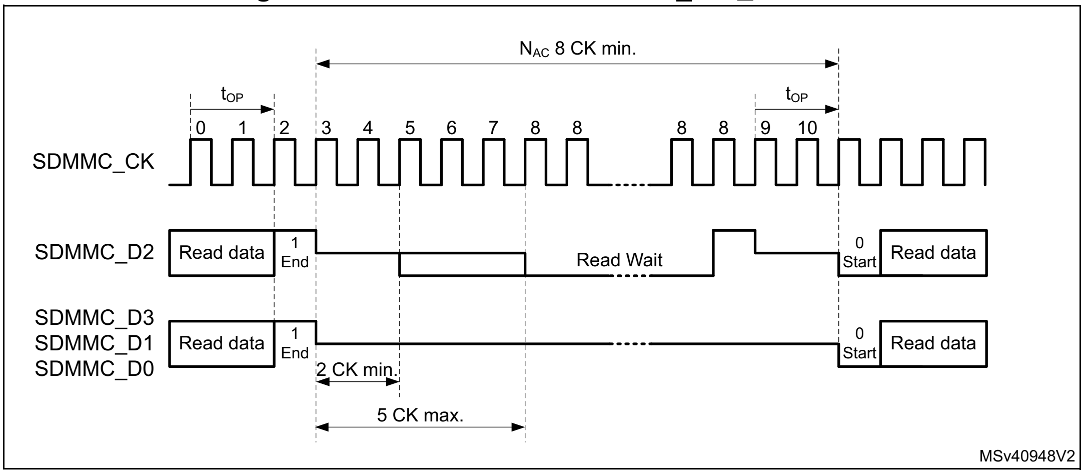
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 670*.

Figure 670. Read Wait with SDMMC_CK ≤ 50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 671*).

Figure 671. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

56.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 425. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

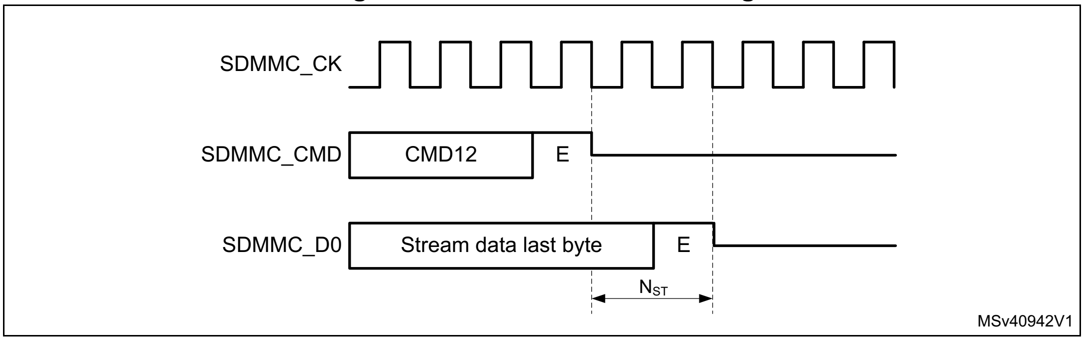
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   – This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   – The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
     The card when no data is being sent, does not start sending any new data.
   – The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
     The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   – When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   – When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   – When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   – When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   – When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   – When DATALENGHT < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 672. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   – CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   – The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   – CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   – CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

56.6.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

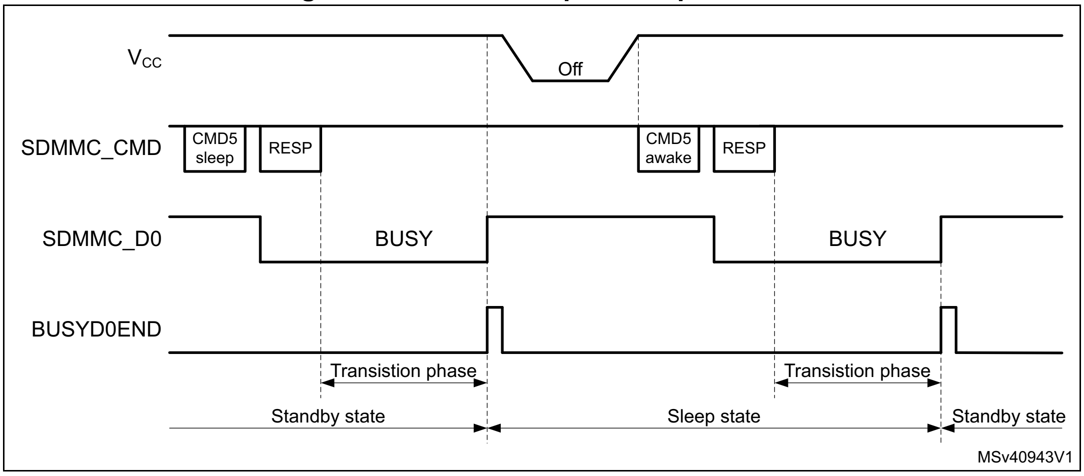
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 673. CMD5 Sleep Awake procedure



56.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   – This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   – Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   – To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

56.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

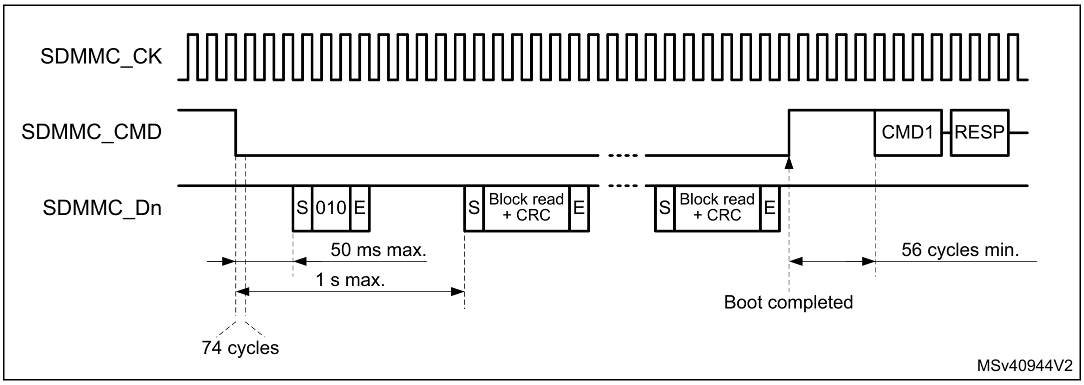
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 674. Normal boot mode operation



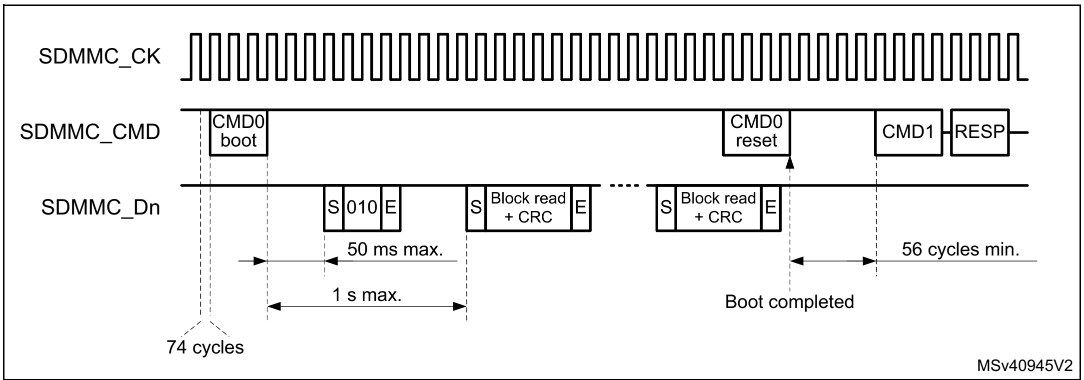
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   - If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 675. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

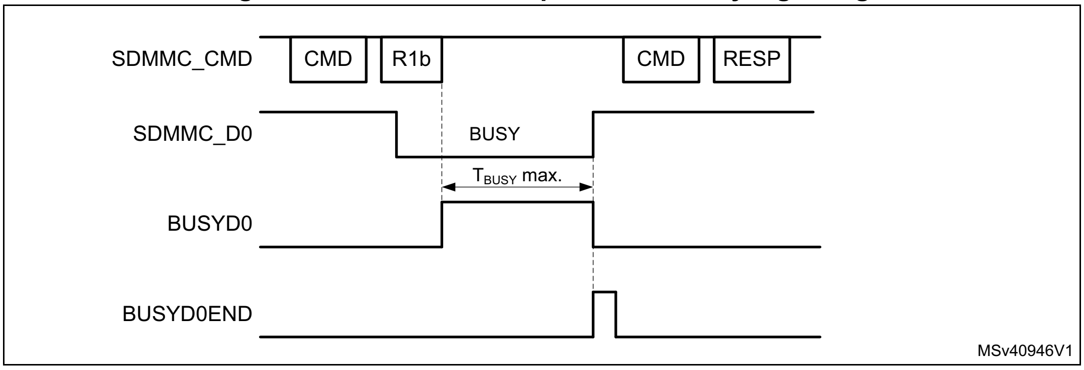
1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   – the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   – the ACK timeout to start.
   – DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   – when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   – when data CRC fails the DCRCFAIL flag is also generated.
   – when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
    – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

56.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 676. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  – If BUSYD0 signals not busy, signal busy release to firmware
  – If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

56.6.7 Reset and card cycle power

Reset

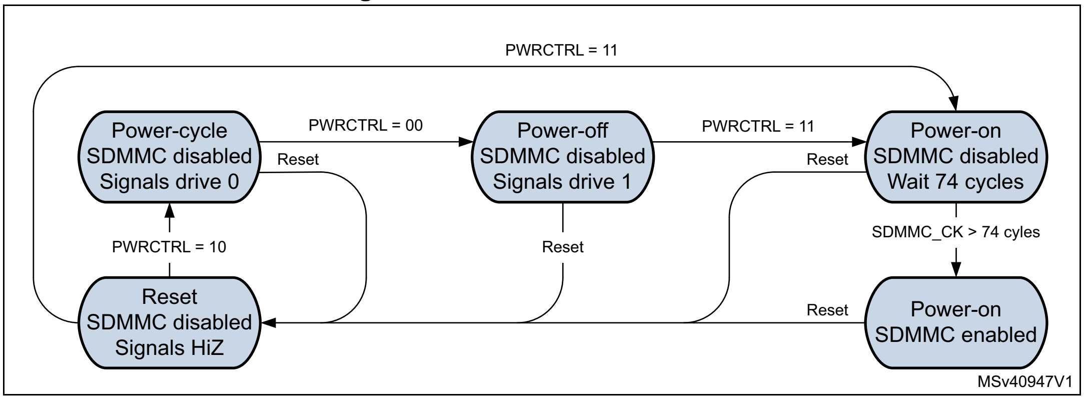
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 677*..

Figure 677. SDMMC state control

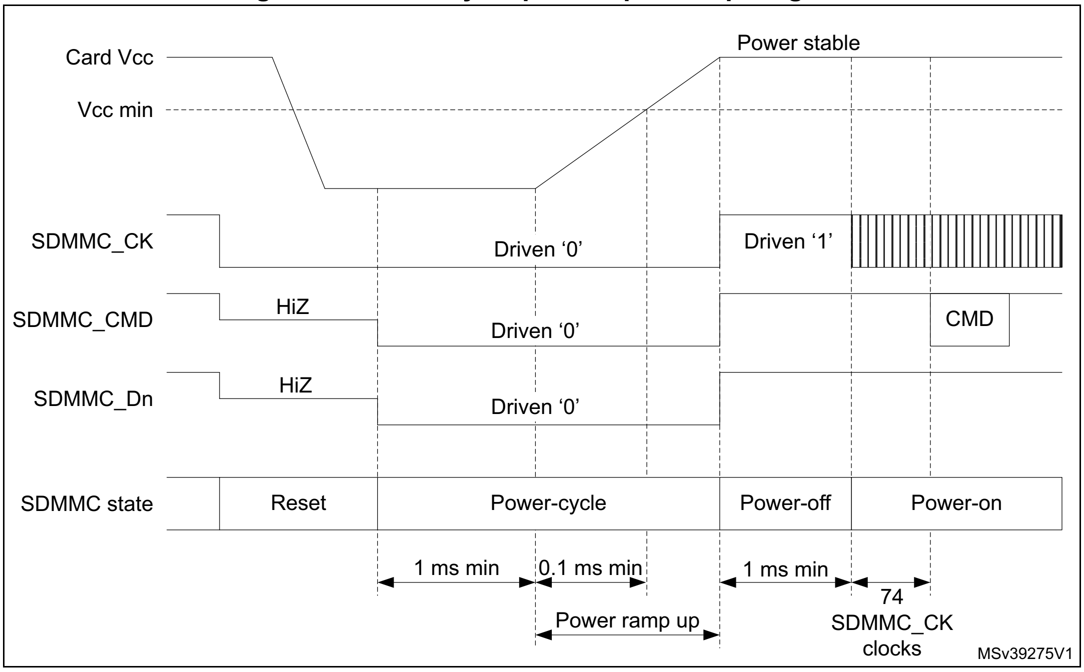


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 678. Card cycle power / power up diagram



56.7 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

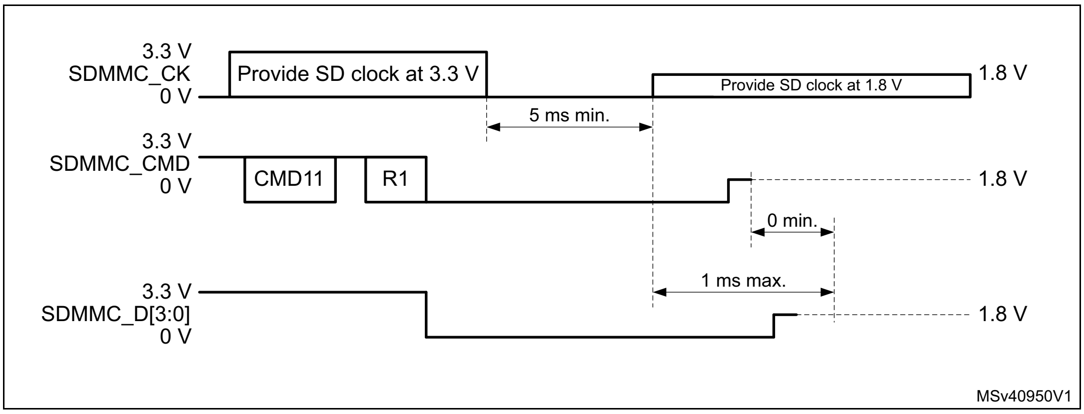
To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control can not be used.

56.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 679. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   – if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   – if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   – When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   – When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

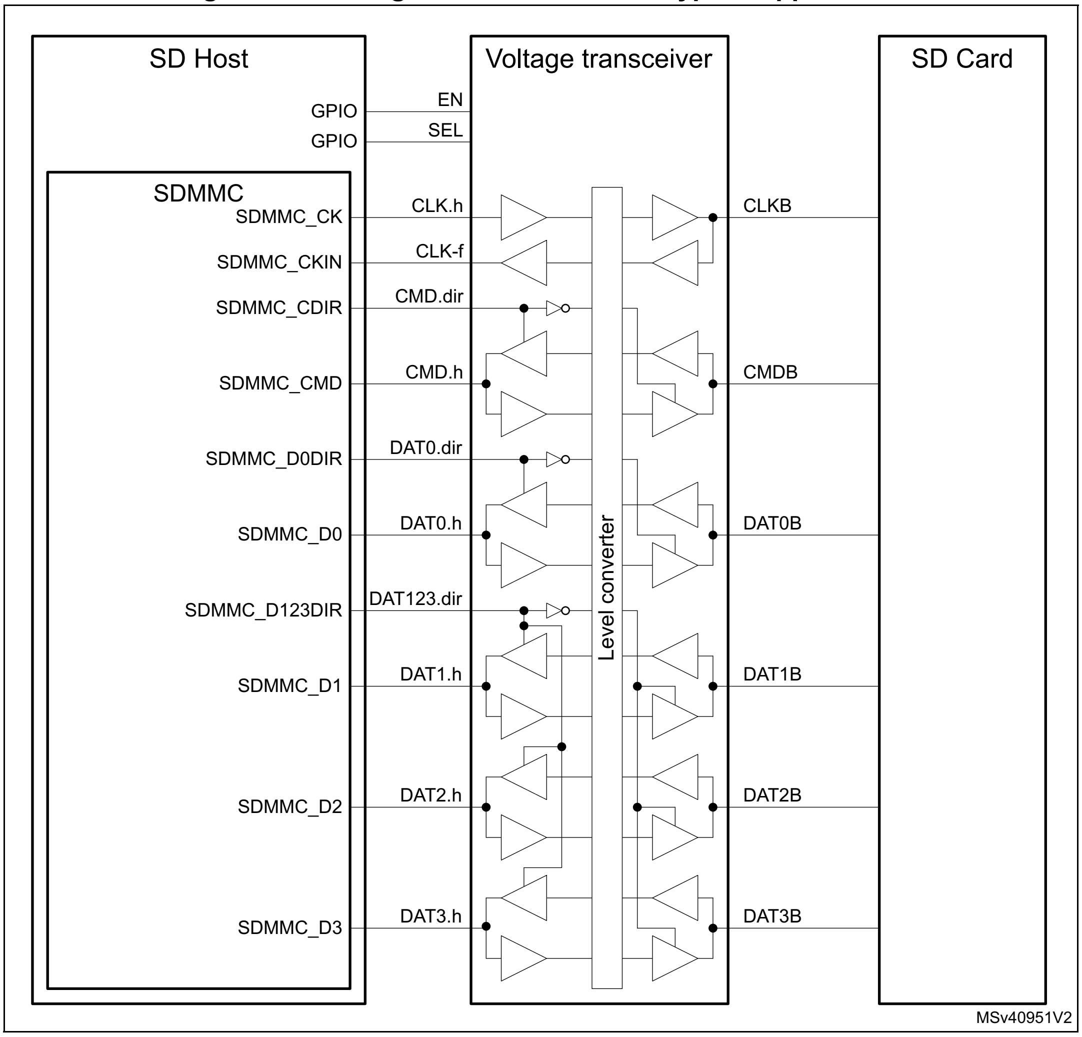
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    - When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    - When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.

Figure 680. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

56.9 SDMMC interrupts

Table 426. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	N/A	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	N/A	Yes
SDMMC	Transmit FIFO full	TXFIFOF	N/A	N/A	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	N/A	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	N/A	Yes
SDMMC	Receive FIFO empty	RXFIFOE	N/A	N/A	Yes
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

56.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHB bus error.

56.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver I/Os driven as output when direction signal is low.

1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

56.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC _EN	NEG EDGE
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division > 1 (CLKDIV > 0).

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in *Section 56.10.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:
- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:
- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:
- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:
- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: ..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: ..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

56.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMDARG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0).

Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

56.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.

CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled

1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)

0: Normal boot mode procedure selected

1: Alternative boot mode procedure selected

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.

If this bit is set, the CPSM is enabled.

When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.

During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.

WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).

If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.

00: No response, expect CMDSENT flag

01: Short response, expect CMDREND or CCRCFAIL flag

10: Short response, expect CMDREND flag (No CRC)

11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

56.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.

56.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CARDSTATUS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTATUS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below

See *Table 427*.

The card status size is 32 or 128 bits, depending on the response type.

Table 427. Response type and SDMMC_RESPxR registers

Register[1]	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0][2]

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

56.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATATIME[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATATIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

56.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATALENGTH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

56.10.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
0000: Block length = 2^0 = 1 byte
0001: Block length = 2^1 = 2 bytes
0010: Block length = 2^2 = 4 bytes
0011: Block length = 2^3 = 8 bytes
0100: Block length = 2^4 = 16 bytes
0101: Block length = 2^5 = 32 bytes
0110: Block length = 2^6 = 64 bytes
0111: Block length = 2^7 = 128 bytes
1000: Block length = 2^8 = 256 bytes
1001: Block length = 2^9 = 512 bytes
1010: Block length = 2^{10} = 1024 bytes
1011: Block length = 2^{11} = 2048 bytes
1100: Block length = 2^{12} = 4096 bytes
1101: Block length = 2^{13} = 8192 bytes
1110: Block length = 2^{14} = 16384 bytes
1111: Reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)
When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
00: Block data transfer ending on block count.
01: SDIO multibyte data transfer.
10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: From host to card.
1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.
This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or *e*•MMC cards).
0: Do not start data transfer without CPSM data transfer command.
1: Start data transfer without CPSM data transfer command.

56.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:16]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATACOUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

56.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected

This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

0: card SDMMC_D0 signal does NOT signal change from busy to not busy.

1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response

This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.

0: card signals not busy on SDMMC_D0.

1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty

This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty

This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full

This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full

This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full

There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty

At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)

This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)

This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received

DBCKEND is set when:

- CRC check passed and DPSM moves to the R_W state

or

- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly

DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)

Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

56.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

56.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
0: Voltage switch clock stopped interrupt disabled
1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

56.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACKTIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

56.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IDMABMODE**: Buffer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: Single buffer mode.
1: Linked list mode.

Bit 0 **IDMAEN**: IDMA enable

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: IDMA disabled
1: IDMA enabled

56.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMA BNDT [11]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABNDT[10:0]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:5 **IDMABNDT[11:0]**: Number of bytes per buffer

This 12-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x001: buffer size = 8 words = 32 bytes.
Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 Kbyte.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

56.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABASE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IDMABASE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r

Bits 31:0 **IDMABASE[31:0]**: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

56.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IDMALA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: SDMMC_IDMALAR is not to be updated, last linked list item.

1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: SDMMC_IDMABSIZER is not to be updated from next linked list table.

1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.

0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).

1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset

Linked list item offset pointer to the base of the next linked list item structure.

Linked list item base address is IDMABA + IDMALA.

These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

56.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDMABA[29:14]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDMABA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address

Linked list memory base pointer.

These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 1:0 Reserved, must be kept at reset value.

56.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFODATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIFODATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

56.10.21 SDMMC version register (SDMMC_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: SDMMC major revision number

Bits 3:0 **MINREV[3:0]**: SDMMC minor revision number

56.10.22 SDMMC identification register (SDMMC_IPIDR)

Address offset: 0x3F8

Reset value: 0x0014 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IP_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IP_ID[31:0]**: SDMMC identification

56.10.23 SDMMC size identification register (SDMMC_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: SDMMC size identification

56.10.24 SDMMC register map

Table 428. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SDMMC_POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ARGR	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 428. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	SDMMC_CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																
0x010	SDMMC_RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_RESP1R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SDMMC_RESP2R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SDMMC_RESP3R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_RESP4R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SDMMC_DTIMER	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SDMMC_DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SDMMC_DCTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SDMMC_DCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SDMMC_STAR	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SDMMC_ICR	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0

Table 428. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03C	SDMMC_ MASKR	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0
0x040	SDMMC_ ACKTIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044 - 0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	SDMMC_ IDMACTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11:0]												Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0					
0x058	SDMMC_ IDMABASER	IDMABASE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080 + 0x04 * x, (x=0..15)	SDMMC_ FIFORx	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0 - 0x3F0	Reserved	Res.																															
0x3F4	SDMMC_ VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	1	0	0	0	0	0
0x3F8	SDMMC_ IPIDR	IP_ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
0x3FC	SDMMC_ SIDR	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

57 Controller area network with flexible data rate (FDCAN)

57.1 Introduction

The controller area network (CAN) subsystem (see *Figure 681*) consists of two CAN modules, a shared message RAM, and a clock calibration unit. Refer to the product memory organization for the base address of each of them.

FDCAN modules are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

In addition, the first CAN module FDCAN1 supports time triggered CAN (TTCAN), specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. The FDCAN1 contains additional registers, specific to the time triggered feature. The CAN FD option can be used together with event-triggered and time triggered CAN communication.

A 10-Kbyte message RAM implements filters, receive FIFOs, receive buffers, transmit event FIFOs, transmit buffers (and triggers for TTCAN). This memory is shared between the FDCAN modules.

The common clock calibration unit is optional. It can be used to generate a calibrated clock for each FDCAN from the HSI internal RC oscillator and the PLL, by evaluating CAN messages received by the FDCAN1.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 429* and *Table 430*.

Table 429. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_cal_it	Digital output	FDCAN calibration interrupt
fdcan1_intr0_it	Digital output	FDCAN1 interrupt0
fdcan1_intr1_it		FDCAN1 interrupt1
fdcan2_intr0_it	Digital output	FDCAN2 interrupt0
fdcan2_intr1_it		FDCAN2 interrupt1
fdcan1_swt[0:5]	Digital input	Stop watch trigger input
fdcan1_evt[0:5]		Event trigger input
fdcan1_ts[0:15]		External timestamp vector (coming from tim3_counter[15:0])
fdcan1_soc	Digital output	Start of cycle pulse
fdcan1_rtp		Register time mark pulse
fdcan1_tmp		Trigger time mark pulse

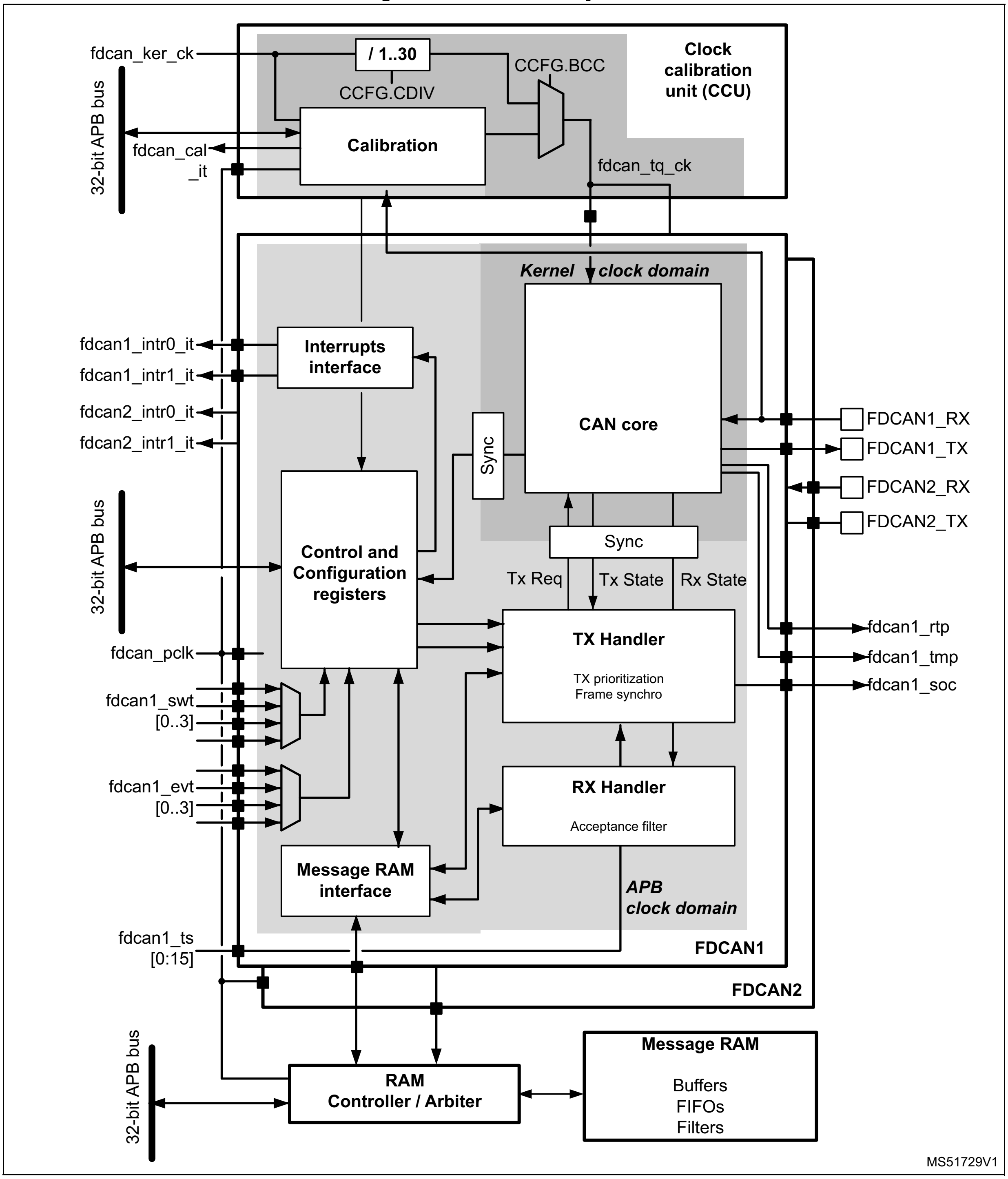
Table 430. CAN subsystem I/O pins

Name	Type	Description
FDCAN1_RX	Digital input	FDCAN1 receive pin
FDCAN1_TX	Digital output	FDCAN1 transmit pin
FDCAN2_RX	Digital input	FDCAN2 receive pin
FDCAN2_TX	Digital output	FDCAN2 transmit pin

Table 431. CAN triggers

Type	ID	Source	Destination
Internal	SWT0	TIM2_TRGO	-
	SWT1	TIM3_TRGO	-
	SWT2	ETH1_PPS	-
	SWT3	-	-
	EVT0	TIM2_TRGO	-
	EVT1	TIM3_TRGO	-
	EVT2	ETH1_PPS	-
	EVT3	-	-
External	fdcan_tt_soc	-	TIM5_ITR6
	fdcan_tt_tmp	-	TIM5_TI1_1, ETH1_PTP3
	fdcan_tt_rtp	-	TIM5_TI1_2

Figure 681. CAN subsystem



57.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015, -4
- CAN FD with max. 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware (FDCAN1 only)
- Event synchronized time-triggered communication supported (FDCAN1 only)
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two configurable receive FIFOs
- Separate signaling on reception of high priority messages
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Configurable transmit FIFO / queue
- Configurable transmit event FIFO
- FDCAN modules share the same message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

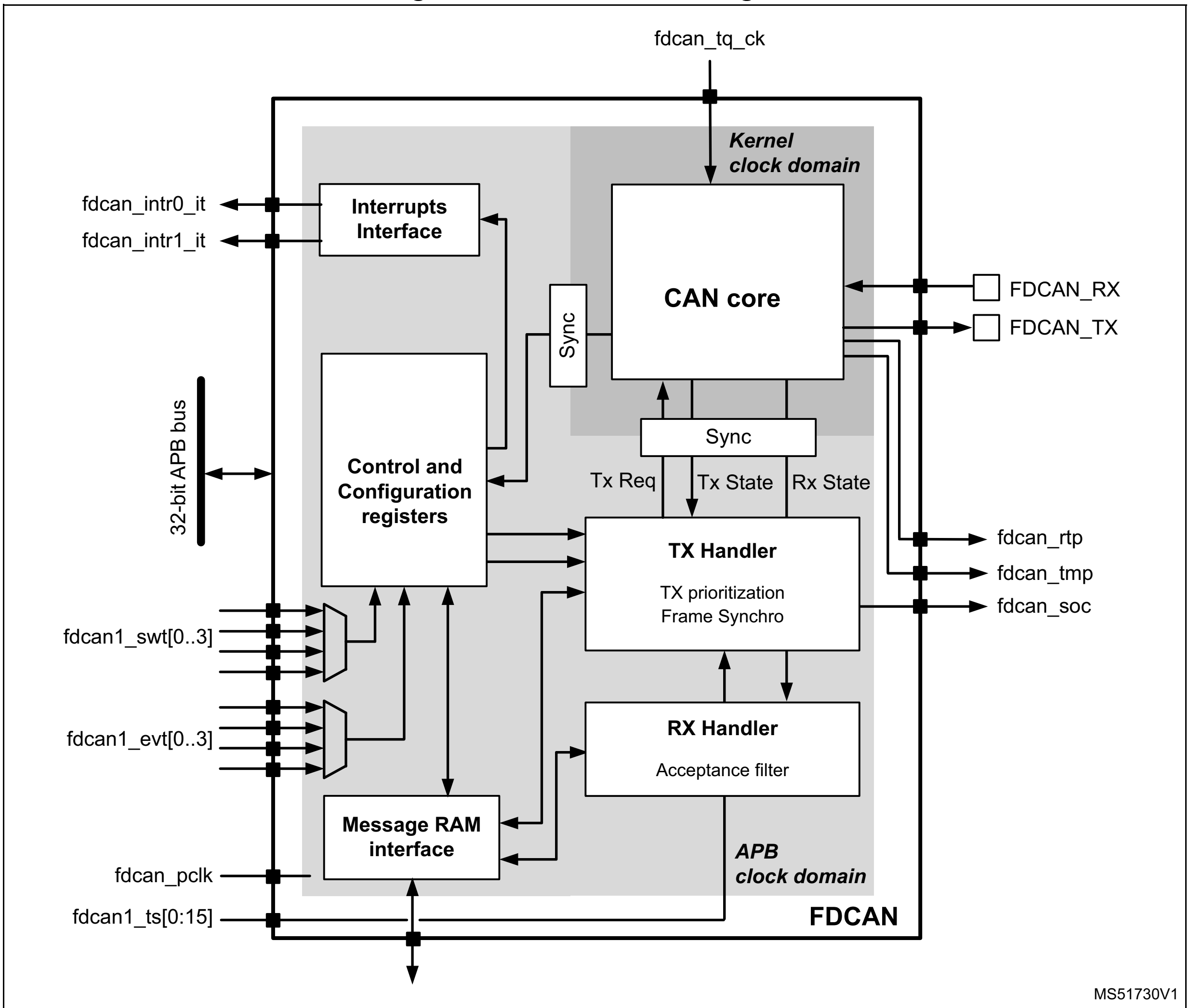
57.3 FDCAN implementation

Table 432. Main features

Module	FDCAN1	FDCAN2
TTCAN	X	-

57.4 FDCAN functional description

Figure 682. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it. By programming EINT0 and EINT1 bits in the FDCAN_ILE register, the interrupt lines can be enabled or disabled separately.

CAN core

The CAN core contains the protocol controller and the receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions, and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain, and vice versa.

Tx handler

Controls the message transfer from the message RAM to the CAN core. A maximum of 32 Tx buffers can be configured for transmission. Tx buffers can be used as dedicated Tx buffers, as Tx FIFO, part of a Tx queue, or as a combination of them. A Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

On FDCAN1, the Tx handler also implements the frame synchronization entity (FSE) which controls time-triggered communication according to ISO11898-4. It synchronizes itself with the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the message RAM. Stop watch trigger, event trigger, and time mark interrupt are synchronization interfaces.

Rx handler

Controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, each of configurable size, and up to 64 dedicated Rx buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx buffer, in contrast to a receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

APB interface

Connects the FDCAN to the APB bus.

Message RAM interface

Connects the FDCAN access to an external 10 Kbytes message RAM through a RAM controller/arbiter.

57.4.1 Operating modes

Software initialization

Software initialization is started by setting the INIT bit in the FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While the INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The counters of the error management logic (EML) are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards, the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in the FDCAN_CCCR register and the CCE bit in the FDCAN_CCCR register are set.

The CCE bit in the FDCAN_CCCR register can be set/cleared only while the INIT bit in the FDCAN_CCCR is set. The CCE bit in FDCAN_CCCR register is automatically cleared when the INIT bit in FDCAN_CCCR is cleared.

The following registers are reset when the CCE bit in the FDCAN_CCCR register is set:

- FDCAN_HPMS - high priority message status
- FDCAN_ RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status
- FDCAN_TTOST - TT (time trigger) operation status (FDCAN1 only)
- FDCAN_TTLGT - TT local and global time, only global time FDCAN_TTLGT.GT is reset (FDCAN1 only)
- FDCAN_TTCTC - TT cycle time and count (FDCAN1 only)
- FDCAN_TTCSM - TT cycle sync mark (FDCAN1 only)

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit in FDCAN_CCCR is set.

The following registers can be written only when the CCE bit in the FDCAN_CCCR register is cleared:

- FDCAN_TXBAR - Tx buffer add request
- FDCAN_TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can be set only by software while both INIT bit and CCE bit in FDCAN_CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can be set/cleared only while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

Normal operation

The FDCAN1 default operating mode after hardware reset is event-driven CAN communication without time triggers (FDCAN_TTOCF.OM = 00). It is required that both INIT bit and CCE bit in the FDCAN_CCCR register are set before the TT operation mode can be changed.

Once the FDCAN is initialized and the INIT bit in the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the CAN FD protocol. The first is the long frame mode (LFM), where the data field of a CAN frame can be longer that eight bytes. The second variant is the fast frame mode (FFM), where the control, data, and CRC fields of a CAN frame are transmitted with a higher bitrate than the beginning and the end of the frame. Fast frame mode can be used in combination with long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are now decoded as FDF bit. FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bitrate inside this CAN FD frame is switched. A CAN FD bitrate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for protocol expansions. If the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting bit FDCAN_PSR.PXE. When protocol exception handling is enabled (FDCAN_CCCR.PXHD = 0), this causes the operation state to change from receiver (FDCAN_PSR.ACT = 10) to integrating (FDCAN_PSR.ACT = 00) at the next sample point. In case protocol exception handling is disabled (FDCAN_CCCR.PXHD = 1), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming FDCAN_CCCR.FDOE. If FDCAN_CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of classic CAN frames is always possible. Whether a CAN FD or a classic CAN frame is transmitted, it can be configured via bit FDF in the respective Tx buffer element. With FDCAN_CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted, even if bit FDF of a Tx buffer element is set. FDCAN_CCCR.FDOE and FDCAN_CCCR.BRSE can only be changed while FDCAN_CCCR.INIT and FDCAN_CCCR.CCE are both set.

With FDCAN_CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in classic CAN format. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 1, transmission of CAN FD frames with bitrate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bitrate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bitrate switching option for transmissions.
- During system startup all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking must be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has been completed. Then all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. DLC codes 0 to 8 have the same coding as in standard CAN, codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 433*.

Table 433. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bitrate switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the bit timing and prescaler register FDCAN_NBTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the fast bit timing and prescaler register FDCAN_DBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bitrate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bitrate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long frames and CAN FD fast frames, the value of the bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for CAN FD transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives an FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx buffer or Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bitrate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bitrates during the CAN FD data phase, independent from the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit FDCAN_DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by FDCAN_TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of minimum time quanta (mtq, that is, one period of fdcan_tq_ck clock).

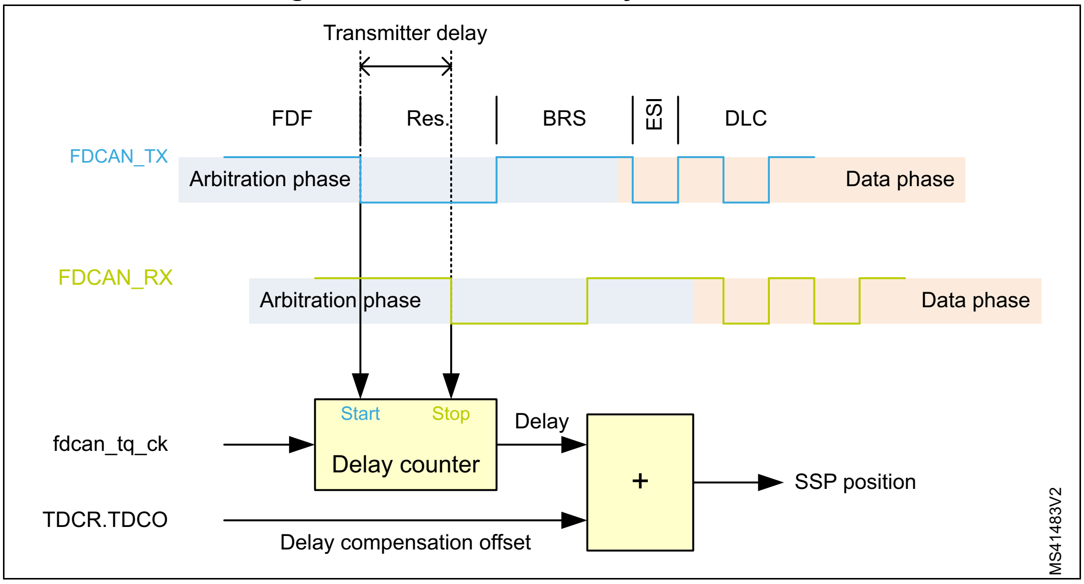
FDCAN_PSR.TDCV shows the actual transmitter delay compensation value, cleared when FDCAN_CCCR.INIT is set, and updated at each transmission of an FD frame while FDCAN_DBTP.TDC is set.

The following boundary conditions must be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCANx_TX to FDCANx_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO must be less than six bit times in the data phase.
- The sum of the measured delay from FDCANx_TX to FDCANx_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO must be less than or equal to 127 mtq. If this sum exceeds 127 mtq, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs

If transmitter delay compensation is enabled by programming FDCAN_DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is 1 mtq.

Figure 683. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming FDCAN_TDCR.TDCF. This defines a minimum value for the SSP position.

Dominant edges on FDCANx_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least FDCAN_TDCR.TDCF and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, but waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (FDCAN_ECR.REC, FDCAN_ECR.TEC) are frozen while error logging (FDCAN_ECR.CEL) is active. The software can set the FDCAN into restricted operation mode by setting bit FDCAN_CCCR.ASM. The bit can be set only by software when both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler was not able to read data from the message RAM in time. To leave restricted operation mode, the software must reset FDCAN_CCCR.ASM.

The restricted operation mode can be used in applications that adapt themselves to different CAN bitrates. In this case the application tests different bitrates and leaves the restricted operation mode after it has received a valid frame.

FDCAN_CCCR.ASM is also controlled by the clock calibration unit. When the calibration process is enabled, the restricted operation mode is entered and the FDCAN_CCR.ASM bit is set. Once the calibration is completed, the FDCAN_CCR.ASM bit is cleared.

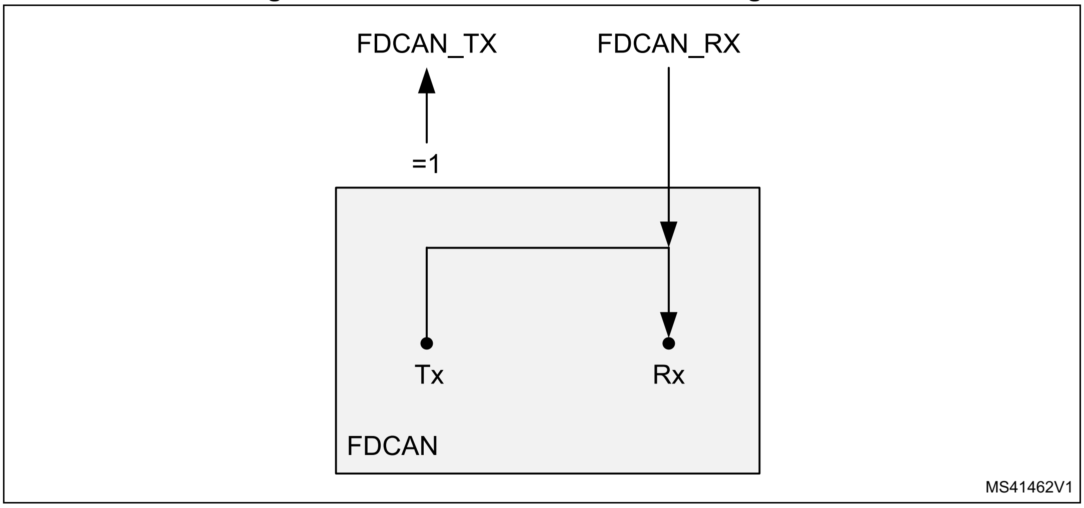
Note: The restricted operation mode must not be combined with the loop back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting FDCAN_CCCR.MON bit or when error level S3 (FDCAN_TTOST.EL = 11) is entered. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode register FDCAN_TXBRP is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 684* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 684. Pin control in bus monitoring mode



Disabled automatic retransmission (DAR) mode

According to the CAN Specification (see ISO11898-1, 6.3.3 recovery management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled.

To support time-triggered communication as described in ISO 11898-1: 2015, chapter 9.2, the automatic retransmission may be disabled via FDCAN_CCCR.DAR.

Frame transmission in disabled automatic retransmission (DAR) mode

In DAR mode all transmissions are automatically canceled after they started on the CAN bus. A Tx buffer Tx request pending bit FDCAN_TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set
- Arbitration loss or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx not set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via register FDCAN_CCCR.CSR. As long as the clock stop request is active, bit FDCAN_CCCR.CSR is read as 1.

When all pending transmission requests have been completed, the FDCAN waits until the bus idle state is detected. Then the FDCAN sets FDCAN_CCCR.INIT to 1 to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power down by setting FDCAN_CCCR.CSA to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to FDCAN_CCCR.INIT has no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application must turn on the module clocks before resetting the CC control register flag FDCAN_CCCR.CSR. The FDCAN acknowledges this by resetting FDCAN_CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit FDCAN_CCCR.INIT.

Test modes

To enable write access to FDCAN test register (see *Section 57.5.4*), bit FDCAN_CCCR.TEST must be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming FDCAN_TEST.TX. Additionally, to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from FDCAN_TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to FDCAN_TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via FDCAN_TEST.RX.

Note: Test modes should be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop back mode

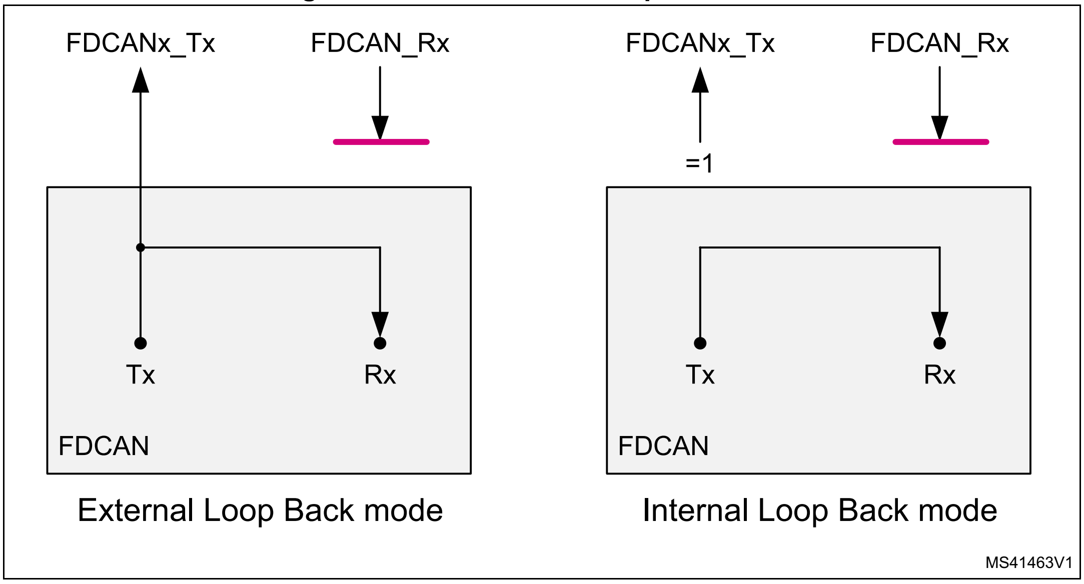
The FDCAN can be set in external loop back mode by programming FDCAN_TEST.LBCK to 1. In loop back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 685* (left side) shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop back mode

Internal loop back mode is entered by programming bits FDCAN_TEST.LBCK and FDCAN_CCCR.MON to 1. This mode can be used for a “Hot Selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, the FDCAN_RX pin is disconnected from the FDCAN and the FDCAN_TX pin is held recessive. *Figure 685* (right side) shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop back mode.

Figure 685. Pin control in loop back mode



Application watchdog (FDCAN1 only)

The application watchdog is served by reading register FDCAN_TTOST. When the application watchdog is not served in time, bit FDCAN_TTOST.AWE is set, all TTCAN communication is stopped, and the FDCAN1 is set into bus monitoring mode.

The TT application watchdog can be disabled by programming the application watchdog limit FDCAN_TTOCF.AWL to 0x00. The TT application watchdog must not be disabled in a TTCAN application program.

Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wraparound counter. A prescaler FDCAN_TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 … 16). The counter is readable via FDCAN_TSCV.TCV. A write access to register FDCAN_TSCV resets the counter to 0. When the timestamp counter wraps around the interrupt flag FDCAN_IR.TSW is set.

On the start of frame reception/transmission the counter value is captured and stored into the timestamp section of an Rx buffer/Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit FDCAN_TSCC.TSS, a 16-bit timestamp can be used.

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as down-counter and uses the same prescaler controlled by FDCAN_TSCC.TCP as the timestamp counter. The timeout counter is configured via register FDCAN_TOCC. The actual counter value can be read from FDCAN_TOCV.TOC. The timeout counter can only be started while FDCAN_CCCR.INIT = 0. It is stopped when FDCAN_CCCR.INIT = 1, for example when the FDCAN enters Bus_Off state.

The operation mode is selected by FDCAN_TOCC.TOS. When operating in Continuous mode, the counter starts when FDCAN_CCCR.INIT is reset. A write to FDCAN_TOCV presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the interrupt flag FDCAN_IR.TOO is set. In Continuous mode, the counter is immediately restarted at FDCAN_TOCC.TOP.

Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented can vary due to the synchronization/re-synchronization mechanism of the CAN core. If the baudrate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

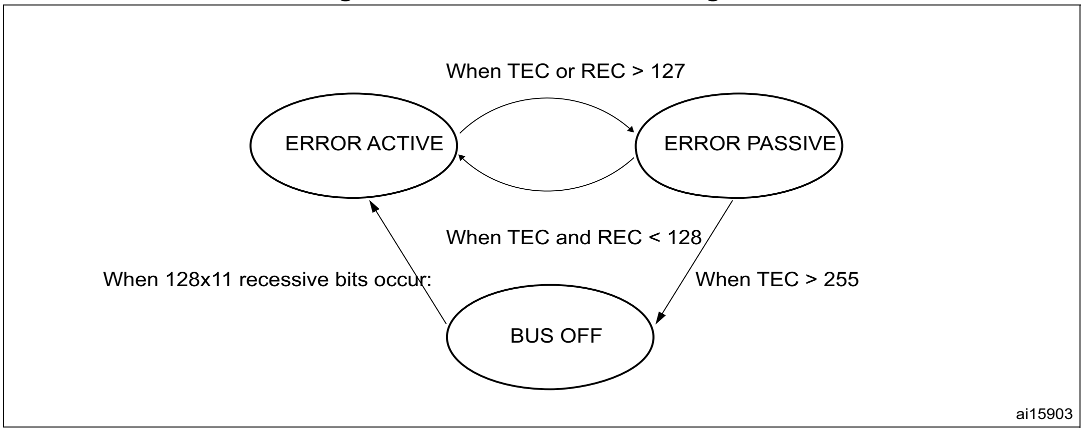
57.4.2 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a transmit error counter (TEC value, in the CAN_ECR register) and a receive error counter (REC value, in the CAN_ECR register), incremented or decremented according to the error condition. Both of them may be read by software to determine the stability of the network. For detailed information about TEC and REC management, refer to the CAN standard.

Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by the BO bit in the CAN_PSR register. In Bus-Off state, the CAN is no longer able to transmit and receive messages. The CAN must wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX).

Figure 686. CAN error state diagram

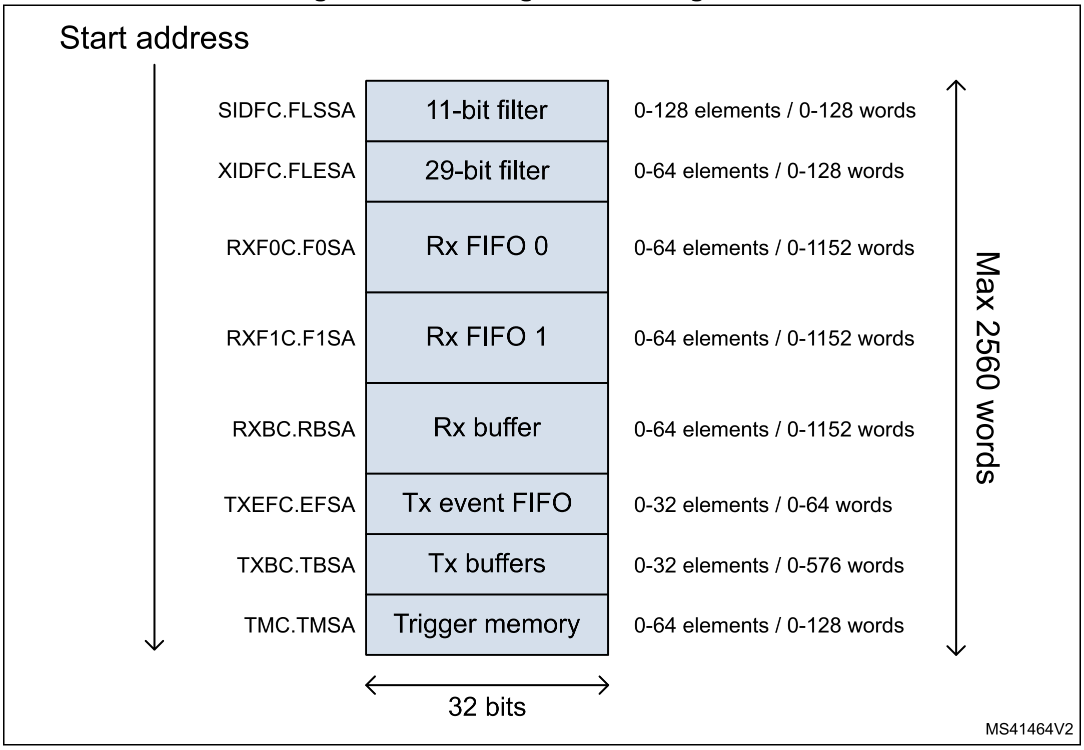


Note: In initialization mode, CAN does not monitor the CAN RX signal, therefore it cannot complete the recovery sequence. To recover, CAN must be in normal mode.

57.4.3 Message RAM

The message RAM has a width of 32 bits. The FDCAN module can be configured to allocate up to 2560 words in the message RAM. It is not necessary to configure each of the sections listed in *Figure 687*, nor is there any restriction with respect to the sequence of the sections.

Figure 687. Message RAM configuration



When the FDCAN addresses the message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The FDCAN does not check for erroneous configuration of the message RAM. In particular, the configuration of the start addresses of the different sections and the number of elements of each section must be done carefully to avoid falsification or loss of data.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx buffers or to 1 of the two Rx FIFOs, as well as the Rx FIFO put and get Indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from 0 to 128 elements for the 11-bit filter and from 0 to 64 for the 29-bit filter)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (FDCAN_GFC)
- Standard ID filter configuration (FDCAN_SIDFC)
- Extended ID filter configuration (FDCAN_XIDFC)
- Extended ID AND Mask (FDCAN_XIDAM)

Depending on the configuration of the filter element (SFEC / EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx buffer
- Store received frame in Rx buffer and generate pulse at filter event pin
- Reject received frame
- Set high priority message interrupt flag FDCAN_IR.HPM
- Set high priority message interrupt flag FDCAN_IR.HPM and store the received frame in FIFO 0 or FIFO 1
- Set high priority message interrupt flag FDCAN_IR.HPM and store the received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (for example CRC error), this message is discarded with the following impact:

- **Rx buffer**
  New data flag of matching Rx buffer is not set, but Rx buffer (partly) overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC.
- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type, see FDCAN_PSR.LEC and FDCAN_PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* must be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independently of the filter(s) used. The result of the acceptance filter process depends strongly upon the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID / SF2ID and EF1ID / EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = 00: The message ID of received frames is AND-ed with the extended ID AND Mask (FDCAN_XIDAM) before the range filter is applied
- EFT = 11: The extended ID AND Mask (FDCAN_XIDAM) is not used for range filtering

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element must be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID / EF1ID is used as message ID filter, while SF2ID / EF2ID is used as filter mask.

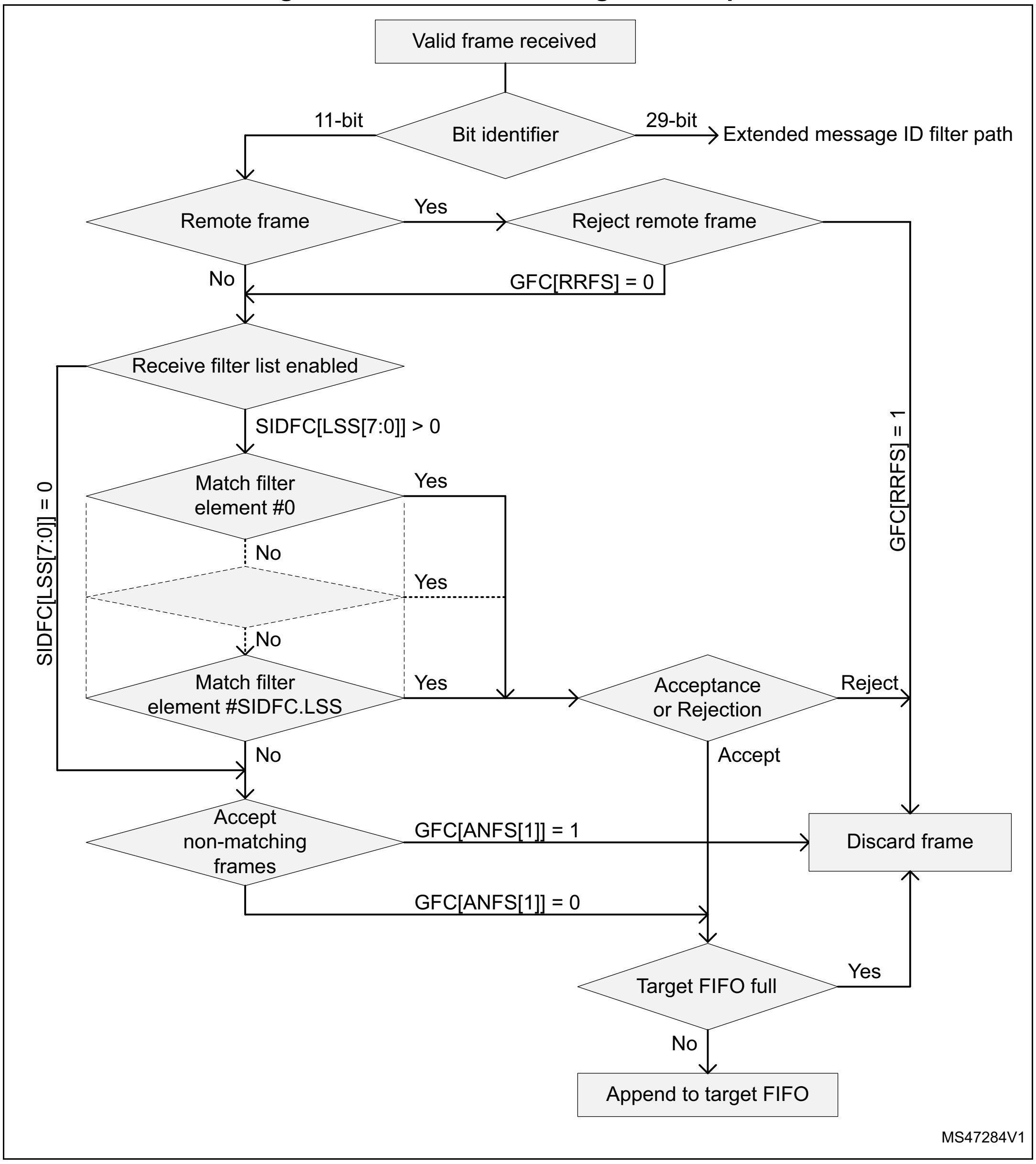
A 0 bit at the filter mask masks out the corresponding bit position of the configured ID filter, for example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are one are relevant for acceptance filtering.

If all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 688 shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 57.4.22*.

Figure 688. Standard message ID filter path

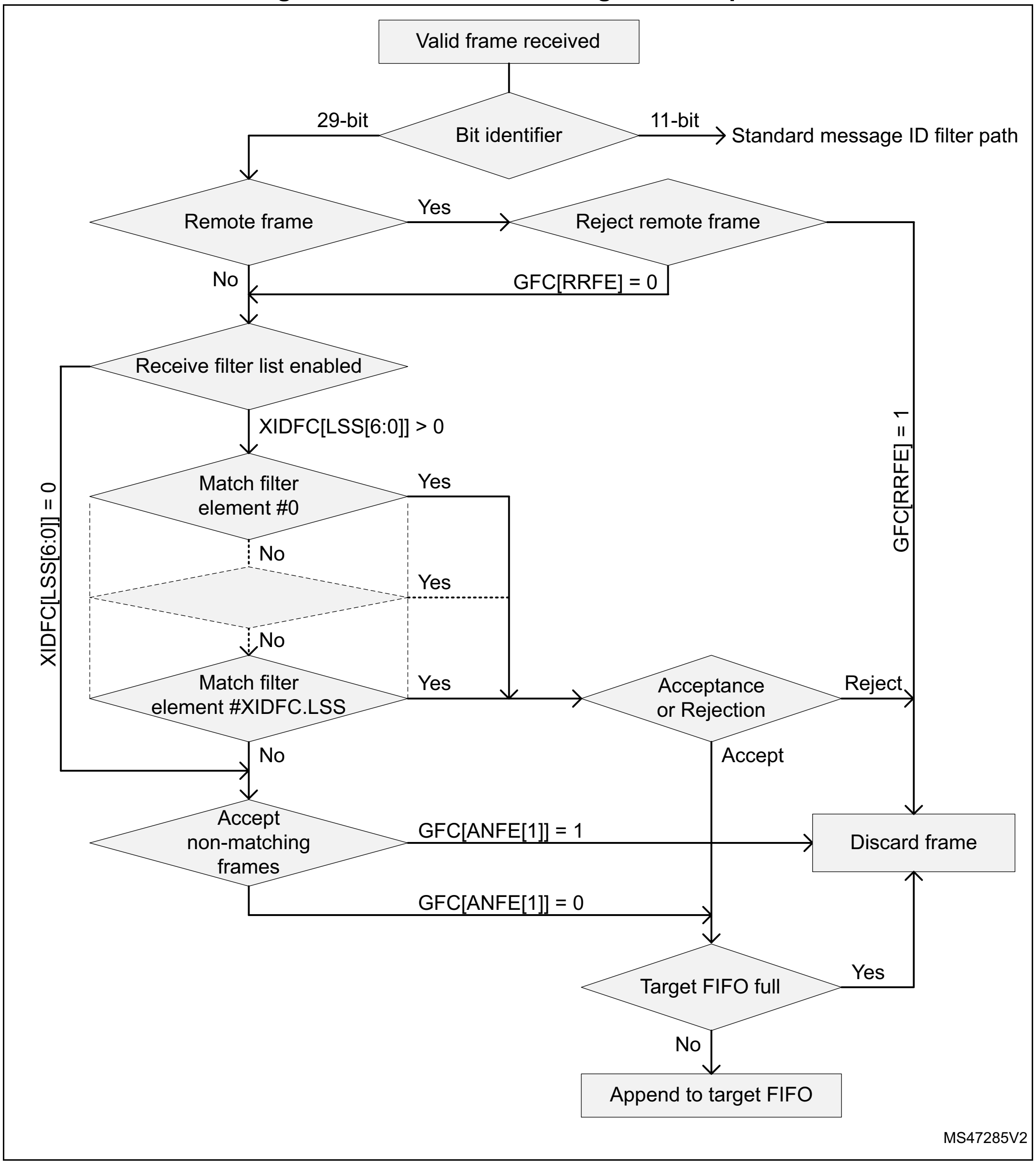


Controlled by the global filter configuration (FDCAN_GFC) and the standard ID filter configuration (FDCAN_SIDFC) message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Extended message ID filtering

Figure 689 shows the flow for extended message ID (29-bit Identifier) filtering. The extended message ID filter element is described in *Section 57.4.23*.

Figure 689. Extended message ID filter path



Controlled by the global filter configuration FDCAN_GFC and the extended ID filter configuration FDCAN_XIDFC message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The extended ID AND Mask (FDCAN_XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers FDCAN_RXF0C and FDCAN_RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx buffer and FIFO element are described in *Section 57.4.19*.

When an Rx FIFO full condition is signaled by FDCAN_IR.RFnF, no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the interrupt flag FDCAN_IR.RFnL is set.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by FDCAN_RXFnC.FnWM, interrupt flag FDCAN_IR.RFnW is set.

When reading from an Rx FIFO, Rx FIFO get index RXFnS[FnGI] + FIFO element size must be added to the corresponding Rx FIFO start address RXFnC[FnSA].

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition, the interrupt flag FDCAN_IR.RFnF is set.

If a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition, the interrupt flag FDCAN_IR.RFnL is set.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is that it can happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 691* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case, the two messages stored in elements 1 and 2 are lost.

After reading from the Rx FIFO, the number of the last element read must be written to the Rx FIFO acknowledge index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

Dedicated Rx buffers

The FDCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via FDCAN_RXBC.RBSA.

For each Rx buffer a standard or extended message ID filter element with SFEC / EFEC=111 and SFID2 / EFID2[10:9] = 00 must be configured (see Section 57.4.22 and Section 57.4.23).

After a received message has been accepted by a filter element, the message is stored into the Rx buffer in the message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition, the flag FDCAN_IR.DRX (message stored in dedicated Rx buffer) in the interrupt register is set.

Table 434. Example of filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective New data flag in register NDAT1,2 is set. As long as the New data flag is set, the respective Rx buffer is locked against updates from received matching frames. The New data flags must be reset by the user by writing a 1 to the respective bit position.

While an Rx buffer New data flag is set, a message ID filter element referencing this specific Rx buffer is not matched, causing the acceptance filtering to continue. The following message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Rx buffer handling

- Reset interrupt flag FDCAN_IR.DRX
- Read New data registers
- Read messages from message RAM
- Reset New data flags of processed messages

Filtering for Debug messages

Filtering for debug messages is done by configuring one standard/extended message ID filter element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC/EFEC must be programmed to 111. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New data flag nor FDCAN_IR.DRX are set. The reception of debug messages can be monitored via FDCAN_RXF1S.DMS.

Table 435. Example of filter configuration for Debug messages

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO.Up to 32 Tx buffers can be set up for message transmission (see *Dedicated Tx buffers*). Depending on the configuration of the element size (FDCAN_RXESC), between two and sixteen 32-bit words (Rn = 3 ... 17) are used for storage of a CAN message data field.

Table 436. Possible configurations for frame transmission

FDCAN_CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bitrate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bitrate switching
1	1	1	1	FD with bitrate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register FDCAN_TXBRP is updated, or when a transmission has been started.

Transmit pause

This feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, after the first successful message transmission, it waits for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit in the FDCAN_CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is disabled (FDCAN_CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmission under complete control of the CPU. Each dedicated Tx buffer is configured with a specific message ID. In case that multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an add request via FDCAN_TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of a dedicated Tx buffer in the message RAM is calculated by adding the element size defined in *Table 437* times the transmit buffer index (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA

Table 437. Tx buffer/FIFO - Queue element size

FDCAN_TXESC.TBDS[2:0]	Data field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Tx FIFO

Tx FIFO operation is configured by programming FDCAN_TXBC.TFQM to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index FDCAN_TXFQS.TFGI. After each transmission the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level FDCAN_TXFQS.TFFL as the difference between get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages must be written to the Tx FIFO starting with the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (FDCAN_TXFQS.TFQF = 1) is signaled. In this case, no further messages should be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via FDCAN_TXBAR. The put index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO free level remain unchanged.

A TX FIFO element allocates element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx FIFO buffer is calculated by adding element size times the put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

Tx queue

Tx queue operation is configured by programming FDCAN_TXBC.TFQM to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). If multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request cyclically increments the put index to the next free Tx buffer. In case that the Tx queue is full (FDCAN_TXFQS.TFQF = 1), the put index is not valid and no further message should be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

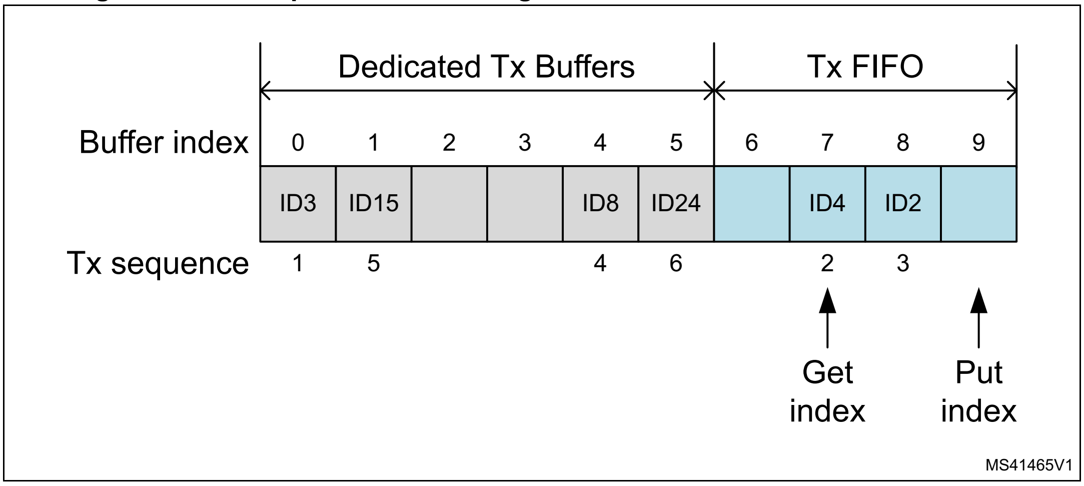
The application may use register FDCAN_TXBRP instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx queue buffer is calculated by adding element size times the Tx queue put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA

Mixed dedicated Tx buffers / Tx FIFO

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx buffers assigned to the Tx FIFO is configured by FDCAN_TXBC.TFQS. In case, FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 690. Example of mixed configuration dedicated Tx buffers / Tx FIFO



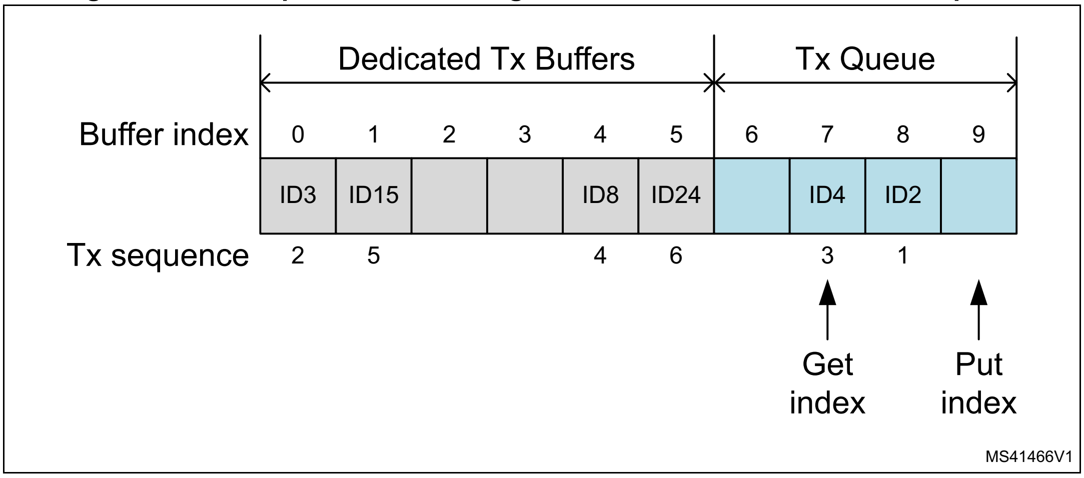
Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by FDCAN_TXFS.TFGI)
- Buffer with lowest message ID gets highest priority and is transmitted next

Mixed dedicated Tx buffers / Tx queue

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx queue buffers is configured by FDCAN_TXBC.TFQS. If FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 691. Example of mixed configuration dedicated Tx buffers / Tx queue



Tx priority setting:

- Scan all Tx buffers with activated transmission request
- Tx buffer with lowest message ID gets highest priority and is transmitted next

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer the user must write a 1 to the corresponding bit position (= number of Tx buffer) of register FDCAN_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register FDCAN_TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN_TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: If a pending transmission is canceled immediately before this transmission starts, a short time window follows where no transmission is started even if another message is pending in this node. This may enable another node to transmit a message that may have a priority lower than that of the second message in this node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, the message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in *Tx FIFO*. Depending on the configuration of the element size (FDCAN_TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message data field.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling, i.e. a Tx buffer holds only the message to transmit, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by FDCAN_IR.TEFF, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the interrupt flag FDCAN_IR.TEFL is set.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by FDCAN_TXEFC.EFWM, interrupt flag FDCAN_IR.TEFW is set.

When reading from the Tx event FIFO, two times the Tx event FIFO get index FDCAN_TXEFS.EFGI must be added to the Tx event FIFO start address FDCAN_TXEFC.EFSA.

57.4.4 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index, see *FDCAN Rx FIFO 0 acknowledge register*

(FDCAN_RXF0A), *FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)*, and *FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)*. Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care must be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the FIFO acknowledge index should not be written because this would set the get index to a wrong position and also alters the FIFO fill level. In this case, some of the older FIFO elements would be lost.

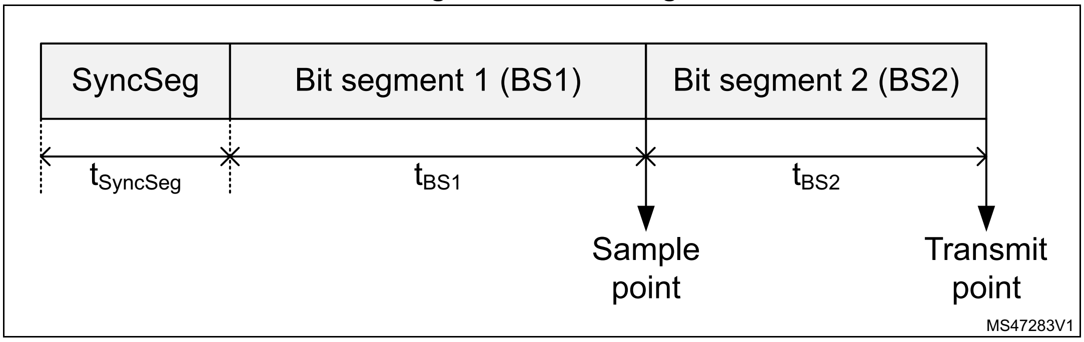
Note: The application must ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

57.4.5 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges. As shown in *Figure 692*, its operation may be explained simply by splitting the bit time in three segments:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, which has a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.

Figure 692. Bit timing



The baudrate is the inverse of bit time (baudrate = 1 / bit time), which, in turn, is the sum of three components. *Figure 692* indicates that bit time = $t_{SyncSeg}$ + t_{BS1} + t_{BS2}, where:

- for the nominal bit time
  - tq = (FDCAN_NBTP.NBRP[8:0] + 1) * $t_{fdcan\_tq\_ck}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_NBTP.NTSEG1[7:0] + 1)
  - t_{BS2} = tq * (FDCAN_NBTP.NTSEG2[6:0] + 1)
- for the data bit time
  - tq = (FDCAN_DBTP.DBRP[4:0] + 1) * $t_{fdcan\_tq\_ck}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_DBTP.DTSEG1[4:0] + 1)
  - t_{BS2} = tq * (FDCAN_DBTP.DTSEG2[3:0] + 1)

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. Registers FDCAN_DBTP and FDCAN_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set.

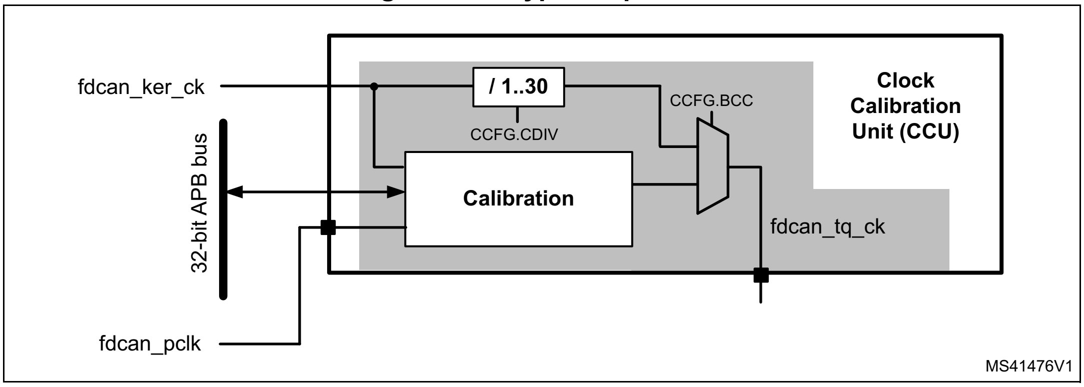
Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

57.4.6 Clock calibration on CAN

After device reset the clock calibration unit (CCU) does not provide a valid clock signal to the FDCAN(s). The CCU must be initialized via FDCAN_CCFG register. The FDCAN_CCFG register can be written only when FDCAN1 has both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set. In consequence the CCU and the FDCAN1 initialization needs to be completed before any FDCAN module can operate.

Clock calibration is bypassed when FDCAN_CCFG.BCC = 1 (see *Figure 693*).

Figure 693. Bypass operation



Operating conditions

The clock calibration on CAN unit is designed to operate under the following conditions:

- a CAN kernel clock frequency fdcan_ker_ck up of at least 80 MHz
- FDCAN bitrates:
  - Nominal bitrate: up to 1 Mbit/s
  - Data bitrate: between nominal bitrate and 8 Mbit/s

The clock calibration on FDCAN unit generates a calibrated time quanta clock fdcan_tq_ck in the range from 0.5 to 25 MHz.

Note: The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock (fdcan_tq_ck < fdcan_pclk). This must be considered when the clock calibration on CAN unit is bypassed (FDCAN_CCFG.BCC = 1).

Calibration accuracy

The calibration accuracy in state Precision_Calibrated depends upon the factors listed below.

- Dynamic clock tolerance at the CAN kernel clock input fdcan_ker_ck
- Measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one fdcan_pclk period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of FDCAN_CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages must be chosen to fit the clock tolerance requirements of the FDCAN1 module.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages for example caused by change of temperature or operating voltage.

Functional description

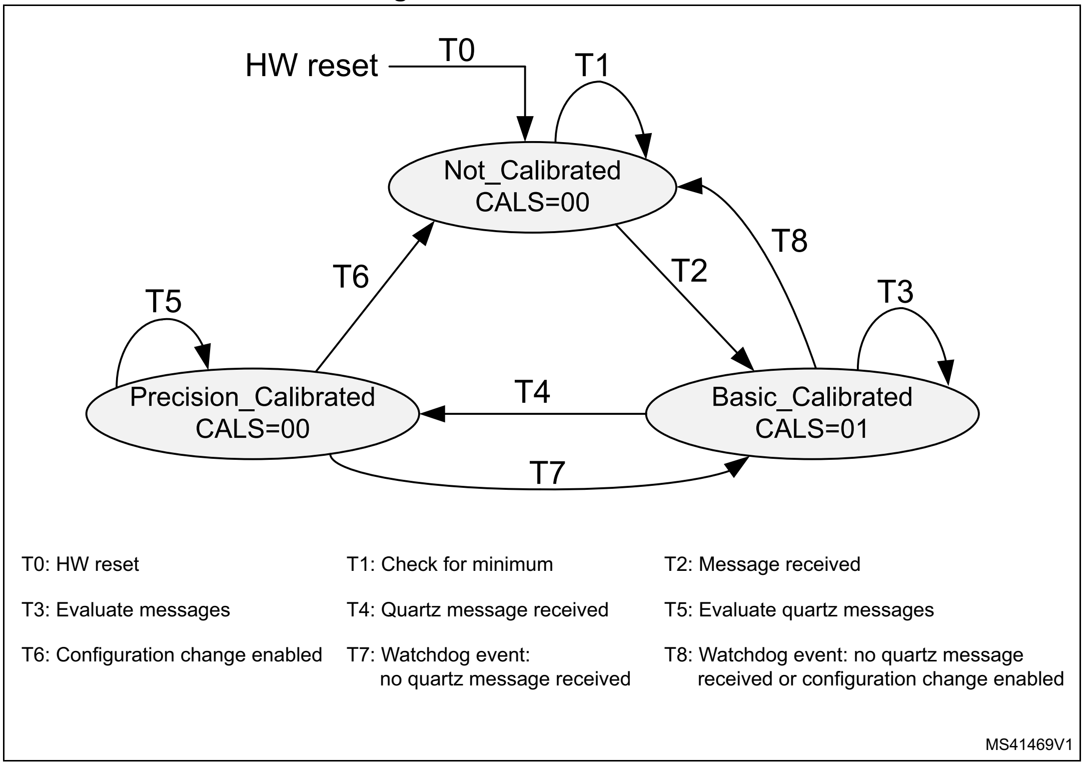
Calibration of the time quanta clock fdcan_tq_ck via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock fdcan_ker_ck.

1. First step: basic calibration
   The minimum distance between two consecutive falling edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider (FDCAN_CCFG.CDIV) is updated each time a new

measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message.

2. Second step: Precision calibration
The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of fdcan_ker_ck periods. The length of this bit sequence can be configured to 32 or 64 bits via FDCAN_CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6-byte data field is required. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 694. FSM calibration



A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC, if enabled by the interrupt enable FDCAN_CCU_IE.CSCE. it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

Until precision calibration is achieved, FDCANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the fdcan_ker_clk is done by software by evaluating the calibration status from register FDCAN_CCU_CSTAT, FDCANs must be set to restricted operation mode (FDCAN_CCCR.ASM = 1) until the calibration on CAN unit is in state Precision_Calibrated (see *Application*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. calibration frames are detected by the FDCAN1 acceptance filtering A filter element and a Rx buffer must be configured in the FDCAN1 to identify and store calibration messages. After reception of a calibration message the Rx buffer new data flag must be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node must transmit its first message after startup with at least one 1010 binary sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the clock calibration on CAN unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the FDCAN1 complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

Configuration

The clock calibration on CAN unit is configured via register FDCAN_CCFG, that is, when FDCAN1 has FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at pin FDCAN1_RX is measured. The number of clock periods depends on the clock frequency applied at input fdcan_ker_ck. In case the measured number of clock periods is below the minimum configured by FDCAN_CCFG.OCPM (as an example, because of a glitch on FDCAN1_RX) the value is discarded and measurement continues.

It is recommended to configure FDCAN_CCFG.OCPM slightly below two CAN bit times:

FDCAN_CCFG.OCPM < ((2 x CAN bit time) / fdcan_ker_ck period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via FDCAN_CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages.

The number of time quanta per bit time configured by FDCAN_CCFG.TQBT is used together with the measured number of oscillator clock periods FDCAN_CCU_CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring FDCAN_CCFG.BCC = 1, the internal clock divider must be configured via FDCAN_CCFG.CDIV to fulfill the condition fdcan_tq_ck < fdcan_pclk.

Note: When clock calibration on CAN is active (FDCAN_CCFG.BCC = 0), the baudrate prescalers of FDCAN modules must be configured to inactive.

Status signaling

The status of the clock calibration on CAN unit can be monitored by reading register FDCAN_CCU_CSTAT. When in state Precision_Calibrated the oscillator clock period counter FDCAN_CCU_CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while FDCAN_CCU_CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by FDCAN_CCU_CSTAT.CALS. A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC. If enabled by the interrupt enable FDCAN_CCU_IE.CSCE it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE (set to high), it remains active until reset by FDCAN_CCU_IE.CWE.

57.4.7 Application

Clock calibration bypassed

The CCU internal clock divider is configured for division by one (FDCAN_CCFG.CDIV = 0x0000). In this operation mode the input clock fdcan_ker_ck is directly routed to the clock output fdcan_tq_ck. In this case fdcan_tq_ck is independent from the configuration and status of FDCAN1and FDCAN2 connected to the CCU. CAN FD operation is possible with an FDCAN_ker_ck above 80 MHz.

Software calibration

The clock calibration on CAN unit also supports software calibration of fdcan_ker_ck by trimming of an on-chip oscillator. For calculation of the trimming values the user must read the CCU state from FDCAN_CCU_CSTAT. The clock from fdcan_ker_ck is routed to output fdcan_tq_ck (FDCAN_CCFG.BCC = 1).

The input clock fdcan_ker_ck must be at least 80 MHz. The clock divider of CCU must be configured via FDCAN_CCFG.CDIV to bring fdcan_tq_ck to a valid range. All other configuration parameters must be set via FDCAN_CCFG. For correct operation of FDCAN1 and FDCAN2, the APB clock fdcan_pclk needs to be equal to or higher than the time quanta clock (fdcan_tq_ck). CAN FD operation is not possible.

For startup FDCAN modules must be both configured for restricted operation (FDCAN_CCCR.ASM = 1) before FDCAN_CCCR.INIT is reset. The input clock fdcan_ker_ck must be adjusted until the clock calibration on CAN unit has reached state Precision_Calibrated. Now the software can reset FDCAN_CCCR.ASM, and the CANFD1 and CANFD2 can start normal operation.

During operation the software must check regularly whether the clock calibration on CAN unit is still in state Precision_Calibrated. In case the clock calibration on CAN unit has left state Precision_Calibrated due to drift of fdcan_ker_ck, FDCAN modules must be set into restricted operation mode by programming FDCAN_CCCR.INIT, FDCAN_CCCR.CCE, and FDCAN_CCCR.ASM to 1. After fdcan_ker_ck has been adjusted successfully (clock calibration on CAN unit is in state Precision_Calibrated), FDCAN modules can resume normal operation.

Note: Trimming accuracy must be to sufficient to meet the CAN clock tolerance requirements for the configured bitrate.

Clock calibration active

This operation mode is entered by resetting FDCAN_CCFG.BCC to 0. In this operation mode the fdcan_ker_ck is controlled by the CCU.

The generation of CCU output signal fdcan_tq_ck depends upon the state of the FDCAN1. Input clock fdcan_ker_ck must be above 80 MHz. Configuration of the CCU and FDCAN1 is required. CAN FD operation is not possible.

If FDCAN1 turns to Bus_Off or when its INIT bit is set by the user command (FDCAN_CCCR.INIT = 1), the CCU enters state Not_Calibrated. CANFD1 and CANFD2 enter restricted operation mode.

Note: This is the default operation mode after reset in case the reset value of FDCAN_CCFG.BCC is configured to 0.

57.4.8 TTCAN operations (FDCAN1 only)

Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the time master (sender of the reference message).

For level 1 the data length must be at least one; for levels 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register FDCAN_TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

Level 1

Level 1 operation is configured via FDCAN_TTOCF.OM = 01 and FDCAN_TTOCF.GEN. external clock synchronization is not available in level 1. The information related to the reference message is stored in the first data byte as shown in *Table 438*. Cycle_Count is optional.

Table 438. First byte of level 1 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					

Level 2

Level 2 operation is configured via FDCAN_TTOCF.OM = 10 and FDCAN_TTOCF.GEN. The information related to the reference message is stored in the first four data bytes as shown in *Table 439*. Cycle_Count and the lower four bits of FDCAN_NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 439. First four bytes of level 2 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

Level 0

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. External event-synchronized time-triggered operation is not available in level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In level 0 Next_is_Gap is always 0. Cycle_Count and the lower four bits of NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 440. First four bytes of level 0 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

57.4.9 TTCAN configuration

TTCAN timing

The network time unit (NTU) is the unit in which all times are measured. The NTU is a constant of the whole network and is defined as a priority by the network system designer. In TTCAN level 1 the NTU is the nominal CAN bit time. In TTCAN level 0 and level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN level 0 and level 2 the length of the NTU is defined by the time unit Ratio TUR. The TUR is in general a non-integer number, given by
TUR = FDCAN_TURNA.NAV / FDCAN_TURCF.DC. The NTU length is given by
NTU = CAN clock period x TUR.

The TUR numerator configuration NC is an 18-bit number, FDCAN_TURCF[NCL[15:0]] can be programmed in the range 0x0000 to 0xFFFF. FDCAN_TURCF[NCH[17:16]] is hard wired to 0b01. When 0xnnnn is written to FDCAN_TURCF[NCL[15:0]], FDCAN_TURNA.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR denominator configuration FDCAN_TURCF.DC is a 14-bit number. FDCAN_TURCF.DC may be programmed in the range 0x0001 to 0x3FFF (0x0000 is an illegal value).

In level 1, NC must be equal to or higher than 4 x FDCAN_TURCF.DC. In level 0 and level 2 NC must be equal to or higher than 8 x FDCAN_TURCF.DC to get the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets FDCAN_TURCF.DC to 0x1000 and FDCAN_TURCF.NCL to 0x10000, resulting in an NTU consisting of sixteen CAN clock periods. Local time and application watchdog are not started before either the FDCAN_CCCR.INIT is reset, or FDCAN_TURCF.ELT is set. FDCAN_TURCF.ELT may not be set before the NTU is configured. Setting FDCAN_TURCF.ELT to 1 also locks the write access to register FDCAN_TURCF.

At startup FDCAN_TURNA.NAV is updated from NC (= FDCAN_TURCF.NCL + 0x10000) when FDCAN_TURCF.ELT is set. In TTCAN level 1 there is no drift compensation. FDCAN_TURNA.NAV does not change during operation, it is always equal to NC.

In TTCAN level 0 and level 2 there are two possibilities for FDCAN_TURNA.NAV to change. When operating as time slave or backup time master, and when FDCAN_TTOCF.ECC is set, FDCAN_TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization, it returns to NC. When operating as the actual time master, and when FDCAN_TTOCF.EECS is set, the user may update FDCAN_TURCF.NCL. When the user sets FDCAN_TTOCN.ECS, FDCAN_TURNA.NAV is updated from the new value of NC at the next reference message. The status flag FDCAN_TTOST.WECS as is set when FDCAN_TTOCN.ECS is set and is cleared when FDCAN_TURNA.NAV is updated. FDCAN_TURCF.NCL is write locked while FDCAN_TTOST.WECS is set.

In TTCAN level 0 and level 2 the clock calibration process adapts FDCAN_TURNA.NAV in the range of the synchronization deviation limit SDL of NC ± 2(FDCAN_TTOCF.LDSDL + 5). FDCAN_TURCF.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of FDCAN_TURNA.NAV.

The synchronization deviation SD is the difference between NC and FDCAN_TURNA.NAV (SD = | NC - FDCAN_TURNA.NAV |). It is limited by the synchronization deviation limit SDL, which is configured by its dual logarithm FDCAN_TTOCF.LDSDL
(SDL = 2 (FDCAN_TTOCF.LDSDL + 5)) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new basic cycle. When the calculated TURNA[NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and FDCAN_TTIR.GTE is set and FDCAN_TTOSC.QCS is reset, or in case of the Disc_Bit = 1, FDCAN_TTIR.GTD is set.

Table 441. TUR configuration example

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
FDCAN _TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

FDCAN_TTOCN.ECS schedules NC for activation by the next reference message. FDCAN_TTOCN.SGT schedules FDCAN_TTGTP.TP for activation by the next reference message. Setting FDCAN_TTOCN.ECS and FDCAN_TTOCN.SGT requires FDCAN_TTOCF.EECS to be set (external clock synchronization enabled) while the FDCAN is actual time master.

The TTCAN module provides an application watchdog to verity the function of the application program. The user must serve this watchdog regularly, else all CAN bus activity is stopped. The application watchdog limit FDCAN_TTOCF.AWL specifies the number of NTUs between two times the watchdog must be served. The maximum number of NTUs is 256. The application watchdog is served by reading register FDCAN_TTOST. FDCAN_TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, the interrupt flag FDCAN_TTIR.AW is set. For software development, the application watchdog may be disabled by programming FDCAN_TTOCF.AWL to 0x00, see *Section 57.6.3*.

Timing of interface signals

The timing events that cause a pulse at output FDCAN trigger time mark interrupt pulse fdcan1_tmp for more than one instance and fdcan_tmp if only one instance; FDCAN register time mark interrupt pulse fdcan1_rtp for more than one instance and fdcan_rtp if only one instance are generated in the CAN clock domain.

There is a clock domain crossing delay to be considered before the same event is visible in the APB clock domain (when FDCAN_TTIR.TTMI is set or FDCAN_TTIR.RTMI is set). As an example, the signals can be connected to the timing input(s) of another FDCAN node (fdcan_swt/fdcan_evt), in order to automatically synchronize two TTCAN networks.

Output FDCAN start of cycle_fdcan1_soc for more than one instance and cycle_fdcan1_soc if only one instance gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the APB clock domain.

57.4.10 Message scheduling

FDCAN_TTOCF.TM controls whether the TTCAN operates as potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message identifier FDCAN_TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest. There cannot be two nodes in the network using the same master priority. FDCAN_TTRMC.RID is used for recognition of reference messages. FDCAN_TTRMC.RMPS is not relevant for time slaves.

The initial reference trigger offset FDCAN_TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for FDCAN_TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

FDCAN_TTOCF.OM decides whether the node operates in TTCAN level 0, level 1, or level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on level 1 in a level 2 network, but not vice versa. The configuration of the TTCAN operation mode via FDCAN_TTOCF.OM is the last step in the setup. With FDCAN_TTOCF.OM = 00 (event-driven CAN communication), the FDCAN operates according to ISO 11898-1: 2015, without time triggers.With FDCAN_TTOCF.OM = 01 (level 1), the FDCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With FDCAN_TTOCF.OM = 10 (level 2), the TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With FDCAN_TTOCF.OM = 11 (level 0), the FDCAN operates as event-driven CAN but maintains a calibrated global time base as in level 2.

FDCAN_TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in levels 0 and level 2 only) the global clock phase to an external reference.

FDCAN_TTMLM.ENTT in the TT matrix limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger repeat factors must be taken into account.

An inaccurate configuration of FDCAN_TTMLM.ENTT results either in a TxCount Underflow (FDCAN_TTIR.TXU = 1 and FDCAN_TTOST.EL = 01, severity 1), or in a Tx count Overflow (FDCAN_TTIR.TXO = 1 and FDCAN_TTOST.EL = 10, severity 2).

Note: In case the first reference message seen by a node does not have Cycle_Count 0, this node may finish its first matrix cycle with its Tx count resulting in a Tx count underflow condition. As long as a node is in state Synchronizing, its Tx_Triggers do not lead to transmissions.

FDCAN_TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of eight basic cycles FDCAN_TTMLM.CCM would be 7. FDCAN_TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

FDCAN_TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window message, the transmission cannot be started in that time window at all. FDCAN_TTMLM.TXEW must be chosen with respect to the network synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

Trigger memory

The trigger memory is part of the external message RAM to which the TTCAN is connected to (see *Section 57.5.26*). It stores up to 64 trigger elements. A trigger memory element consists of time mark TM, cycle code CC, trigger type TYPE, filter type FTYPE, message number MNR, message status count MSC, time mark event internal TMIN, time mark event external TMEX (see *Section 57.4.24*).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory must be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx handler starts to read the message from the message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx handler must be able to prepare the transmission before the transmit trigger time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mar k is reached. If the reference message is n NTU long, a trigger with a time mark lower than n never becomes active and is treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger time mark. The FDCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialization) be seen at a value lower than n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger time mark + 1 NTU + 1 CAN bit time.

The trigger list is different for all nodes in the CAN FD network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

Trigger types

Tx_Ref_Trigger (TYPE = 0000) and Tx_Ref_Trigger_Gap (TYPE = 0001) cause the transmission of a reference message by a time master. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized
time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Tx_Trigger_Single (TYPE = 0010), Tx_Trigger_Continuous (TYPE = 0011), Tx_Trigger_Arbitration (TYPE = 0100), and Tx_Trigger_Merged (TYPE = 0101) cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit is reset.

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code must be considered when the expected number of Tx_Triggers (FDCAN_TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = 0110) and Watch_Trigger_Gap (TYPE = 0111) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Rx_Trigger (TYPE = 1000) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger must be placed after the end of that message transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger time mark is reached.

Time_Base_Trigger (TYPE = 1001) are used to generate internal/external events depending on the configuration of TMIN and TMEX.

End_of_List (TYPE = 1010 … 1111) is an illegal trigger type, a configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger and Watch_Trigger_Gap.

Restrictions for the node trigger list

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) never become active. The watch triggers themselves do not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) must be set to trigger type End_of_List.

A typical trigger list for a potential time master begins with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave is the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the reference trigger offset FDCAN_TTOCF.IRTO. In case error level 2 is reached (FDCAN_TTOST.EL = 10), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- APB clock frequency
- Speed and waiting time for trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

Example for trigger handling

The example shows how the trigger list is derived from a node system matrix. Assumption is that node A is first time master and has knowledge of the section of the system matrix shown in *Table 442*.

Table 442. System matrix, Node A

Cycle count	Time mark						
	1	2	3	4	5	6	7
0	Tx7	-	-	-	-	TxRef	Error
1	Rx3	-	Tx2, Tx4		-	TxRef	Error
2	-	-	-	-	-	TxRef	Error
3	Tx7	-	Rx5	-	-	TxRef	Error
4	Tx7	-	-	Rx6	-	TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the corresponding number of basic cycles in the system matrix is, respectively, 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by FDCAN_TTMLM.CCM. The cycle code CC is composed of repeat factor (= value of most significant 1) and the number of the first basic cycle in the system matrix (= bit field after most significant 1).

As an example, with a cycle code of 0b0010011 (repeat factor = 16, first basic cycle = 3) and a maximum cycle count of FDCAN_TTMLM.CCM = 0x3F matches occur at cycle counts 3, 19, 35 and 51.

A trigger element consists of time mark TM, cycle code CC, trigger type TYPE, and message number MNR. For transmission MNR references the Tx buffer number (0 ... 31). For reception MNR references the number of the filter element (0 ... 127) that matched during acceptance filtering. Depending on the configuration of the filter type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition, a trigger element can be configured for generation of time mark event internal TMIN, and time mark event external TMEX. The message status count MSC holds the counter value (0 ... 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 443. Trigger list, Node A

Trigger	Time mark TM[15:0]	Cycle code CC [6:0]	Trigger type TYPE [3:0]	Mess No. MNR [6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark 1	0b1000000	Rx_Trigger	3
2	Mark 1	0b1000011	Tx_Trigger_Single	7
3	Mark 3	0b1000001	Tx_Trigger_Merged	2
4	Mark 3	0b1000011	Rx_Trigger	5
5	Mark 4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark 4	0b1000100	Rx_Trigger	6
7	Mark 6	N/A	Tx_Ref_Trigger	0 (Ref)
8	Mark 7	N/A	Watch_Trigger	N/A
9	N/A	N/A	End_of_List	N/A

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from 0 until a trigger with time mark higher than cycle time and with its cycle code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the time mark TM, the action defined by trigger type TYPE and message number MNR is started. There is an error in the configuration when End_of_List is reached.

At mark 6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the watch trigger at mark 7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of configuration errors

A configuration error is signaled via FDCAN_TTOST.EL = 11 (severity 3) when:

- The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.
- The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a

Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

- The FSE of a node with FDCAN_TTOCF.TM=0 (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.
- Any time mark placed inside the Tx enable window (defined by FDCAN_TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.
- A time mark is placed near the time mark of a Tx_Ref_Trigger and the reference trigger offset FDCAN_TTOST.RTO causes a reversal of their sequential order measured in cycle time.

TTCAN schedule initialization

The synchronization to the TTCAN message schedule starts when FDCAN_CCCR.INIT is reset. The TTCAN can operate strictly time-triggered (FDCAN_TTOCF.GEN = 0) or external event-synchronized time-triggered (FDCAN_TTOCF.GEN = 1). All nodes start with cycle time 0 at the beginning of their trigger list with FDCAN_TTOST.SYS = 00 (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode ignore Tx_Ref_Trigger and Watch_Trigger and use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

Time slaves

After configuration, a time slave ignores its Watch_Trigger and Watch_Trigger_Gap if it does not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time becomes invalid, but the node is still able to take part in CAN bus communication (to acknowledge or to send error flags). The first received reference message restarts the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter that counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it assumes a fatal error (FDCAN_TTOST.EL = 11, severity 3), set interrupt flag FDCAN_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN_CCCR.MON set to 1). In the bus monitoring mode it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the fatal error state, the user must set FDCAN_CCCR.INIT = 1. After reset of FDCAN_CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets FDCAN_TTOST.SYS = 01 (Synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

Potential time masters

After configuration, a potential time master starts the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It ignores its Watch_Trigger and Watch_Trigger_Gap when it does not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches

Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time becomes invalid, but the node is still able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting FDCAN_TTIR.IWT re-enables the transmission of reference messages until the next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE is not be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it assumes a fatal error (FDCAN_TTOST.EL = 11, severity 3), sets interrupt flag FDCAN_TTIR.WT, switches off its CAN bus output, and enters the bus monitoring mode (FDCAN_CCCR.MON set to 1). In bus monitoring mode, it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets FDCAN_TTOST.SYS = 01 (synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (FDCAN_TTOST.MS = 11) when it was the transmitter of the last reference message, else it is backup time master (FDCAN_TTOST.MS = 10).

When all potential time masters have finished configuration, the node with the highest time master priority in the network becomes the current time master.

57.4.11 TTCAN gap control

All functions related to gap control apply only when the FDCAN is operated in external event synchronized time-triggered mode (FDCAN_TTOCF.GEN = 1). In this operation mode the FDCAN message schedule may be interrupted by inserting gaps between the basic cycles of the system matrix. All nodes connected to the CAN network must be configured for external event- synchronized time-triggered operation.

During a gap, all transmissions are stopped and the CAN bus remains idle. A gap is finished when the next reference message starts a new basic cycle. A gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = 1 for example gaps are initiated by the current time master.

The current time master has two options to initiate a gap. A gap can be initiated under software control when the application program writes FDCAN_TTOCN.NIG = 1. The Next_is_Gap bit is transmitted as 1 with the next reference message. A gap can also be initiated under hardware control when the application program enables the event trigger input pin fdcan_evt by writing FDCAN_TTOCN.GCS = 1. When a reference message is started and FDCAN_TTOCN.GCS is set, a HIGH level at event trigger pin fdcan_evt sets Next_is_Gap = 1.

As soon as that reference message is completed, the FDCAN_TTOST.WFE bit announces the gap to the time master as well as to the time slaves. The current basic cycle continues until its last time window. The time after the last time window is the gap time.

For the actual time master and the potential time masters, FDCAN_TTOST.GSI is set when the last basic cycle has finished and the gap time starts. In nodes that are time slaves, bit FDCAN_TTOST.GSI remains at 0.

When a potential time master is in synchronization state In_Gap (FDCAN_TTOST.SYS = 10), it has four options to intentionally finish a gap:

1. Under software control by writing FDCAN_TTOCN.FGP = 1.
2. Under hardware control (FDCAN_TTOCN.GCS = 1) an edge from HIGH to LOW at the event-trigger input pin fdcan_evt sets FDCAN_TTOCN.FGP and restarts the schedule.
3. The third option is a time-triggered restart. When FDCAN_TTOCN.TMG = 1, the next register time mark interrupt (FDCAN_TTIR.RTMI = 1) sets FDCAN_TTOCN.FGP and starts the reference message.
4. Finally any potential time master finishes a gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

None of these options can cause a basic cycle to be interrupted with a reference message.

Setting of FDCAN_TTOCN.FGP after the gap time begins starts the transmission of a reference message immediately and thereby synchronizes the message schedule. When FDCAN_TTOCN.FGP is set before the gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there is no gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = 1 in the reference message is ignored, as well as the event-trigger input pin fdcan_evt and the bits FDCAN_TTOCN.NIG, FDCAN_TTOCN.FGP, and FDCAN_TTOCN.TMG.

57.4.12 Stop watch

The stop watch function enables capturing of FDCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first must define local time, cycle time, or global time as the stop watch source via FDCAN_TTOCN.SWS. When FDCAN_TTOCN.SWS is different from 00 and TT interrupt register flag FDCAN_TTIR.SWE is 0, the actual value of the time selected by FDCAN_TTTOCN.SWS is copied into FDCAN_TTCPT.SWV on the next rising / falling edge (as configured via FDCAN_TTOCN.SWP) on stop watch trigger pin fdcan_swt. This sets interrupt flag FDCAN_TTIR.SWE. After the application program has read FDCAN_TTCPT.SWV, it may enable the next stop watch event by resetting FDCAN_TTIR.SWE to 0.

57.4.13 Local time, cycle time, global time, and external clock synchronization

There are two possible levels in time-triggered CAN:

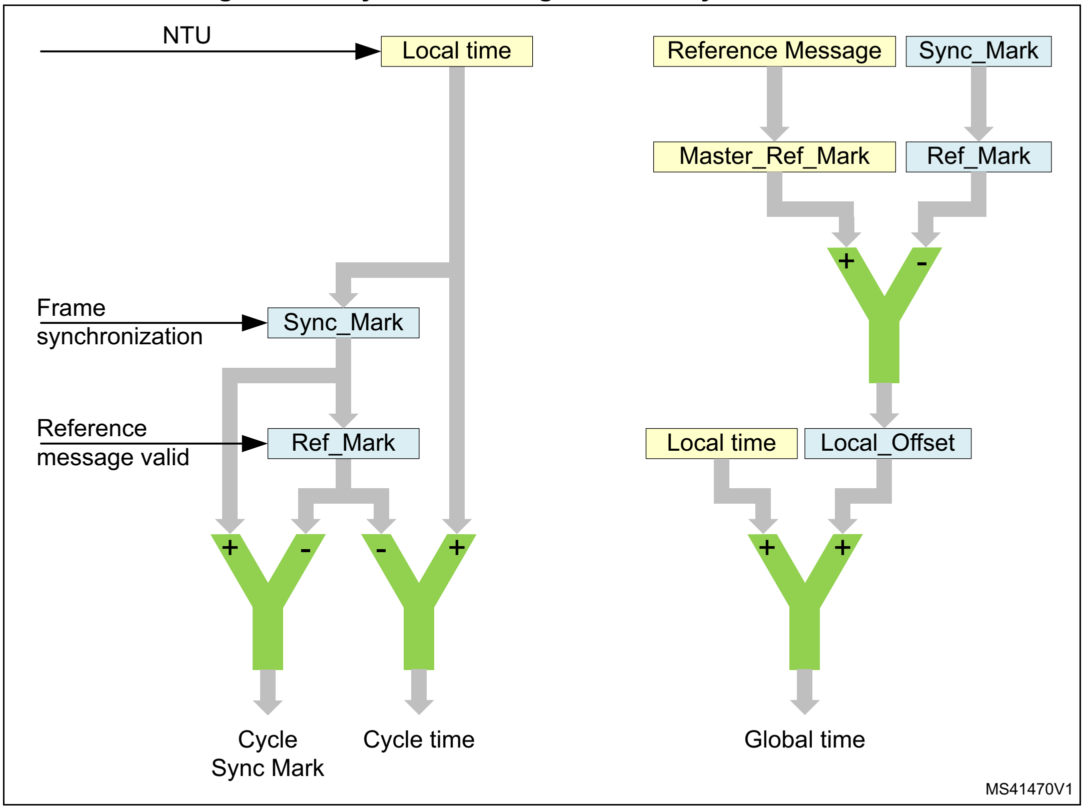
1. Level 1 only provides time-triggered operation using cycle time.
2. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented eight times each NTU. If the length of the NTU is shorter than eight CAN clock periods (as may be configured in level 1, or as a result of clock calibration in level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only four times each NTU.

Figure 695 describes the synchronization of the cycle time and global time, performed in the same manner by all FDCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame synchronization event. This frame synchronization event occurs at the sample point of each start of frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the cycle sync mark (cycle sync mark = Sync_Mark - Ref_Mark) stored in register FDCAN_TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (cycle time = local time - Ref_Mark).

Figure 695. Cycle time and global time synchronization



The cycle time that can be read from FDCAN_TTCTC.CT is the difference of the node local time and Ref_Mark, both synchronized into the APB clock domain and truncated to 16 bit.

The global time exists for TTCAN level 0 and level 2 only, in level 1 it is invalid. The node view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master uses its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3 and 4). The global time that can be read from FDCAN_TTLGT.GT is the sum of the node local time and its local offset, both synchronized into the APB clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see *Figure 696*). The node view of the global time is local time plus local offset. In a potential time master that has never received another time master reference message, Local_Offset is 0. When a node becomes the current time master after first having received other reference messages, Local_Offset is frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register FDCAN_TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

Figure 696. TTCAN level 0 and level 2 drift compensation

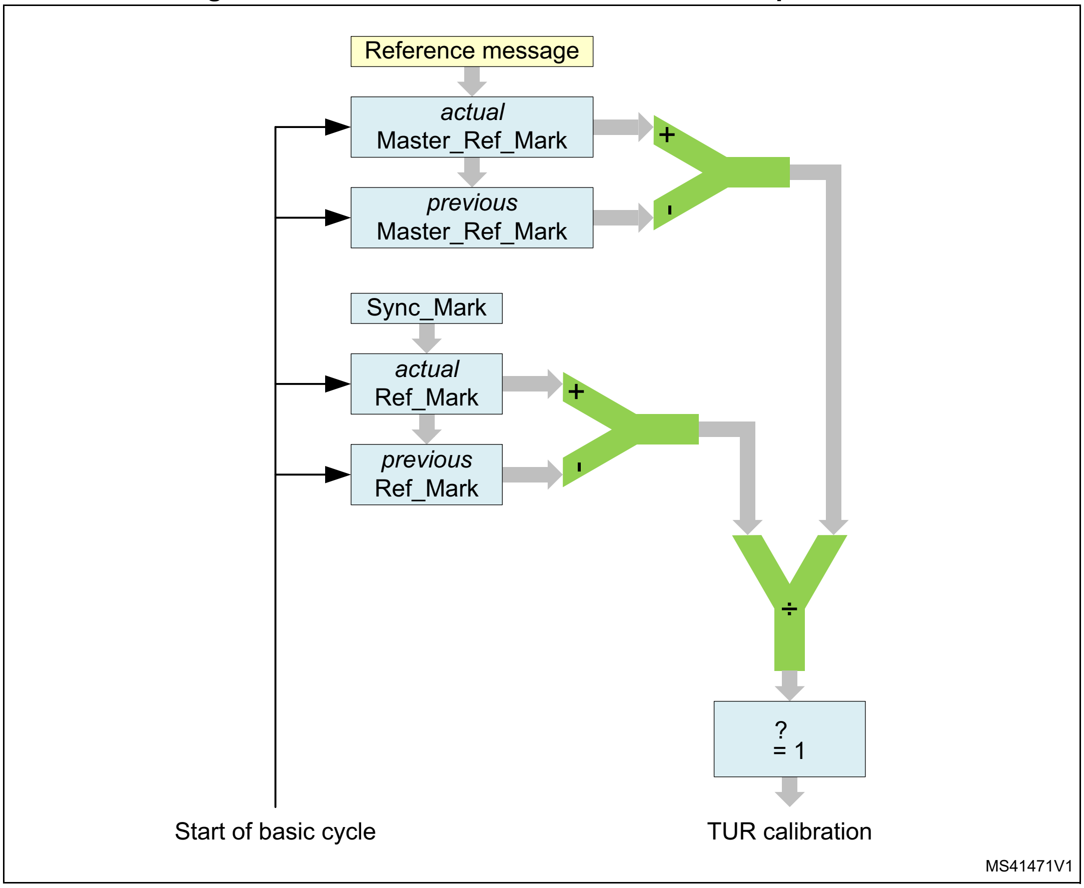


Figure 696 describes how in TTCAN levels 0 and 2 each time receiving node compensates the drift between its own local clock and the time master clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for FDCAN_TURNA.NAV is calculated. If the synchronization deviation SD = | NC - FDCAN_TURNA.NAV | ≤ SDL (synchronization deviation limit), the new value for FDCAN_TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN level 0 and level 2, FDCAN_TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN level 1, FDCAN_TOST.QCS is always 1.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit FDCAN_TTOCF.EECS.

The stop watch function (see *Section 57.4.12*) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated numerator configuration low to

FDCAN_TURCF.NCL (FDCAN_TURCF.DC cannot be updated during operation). The new value takes effect by writing FDCAN_TTOCN.ECS to 1.

The global time phase is adjusted by first writing the phase offset into the TT global time Preset register FDCAN_TTGTP. The new value takes effect by writing FDCAN_TTOCN.SGT to 1. The first reference message transmitted after the global time phase adjustment has the Disc_Bit set to 1.

FDCAN_TTOST.QGTP shows whether the node global time is in phase with the time master global time. FDCAN_TTOST.QGTP is permanently 0 in TTCAN level 1 and when the synchronization deviation limit is exceeded in TTCAN levels 0, 2 (FDCAN_TTOST.QCS = 0). It is temporarily 0 while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = 1 or when FDCAN_TTOST.QCS = 0.

57.4.14 TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

1. S0 - No error
2. S1 - Warning - Only notification of application, reaction application-specific.
3. S2 error - Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the reference trigger offset FDCAN_TTOST.RTO set to the maximum value of 127.
4. S3 - Severe error - Notification of application. All CAN bus operations are stopped, that is, transmission of dominant bits is not allowed, and FDCAN_CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set FDCAN_CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by FDCAN_TTIR.ELC. The error level is monitored by FDCAN_TTOST.EL.

The TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_error (S3)
  – Sets error level FDCAN_TTOST.EL to 11 when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.
- Watch_Trigger_Reached (S3)
  – Sets error level FDCAN_TTOST.EL to 11 when a watch trigger was reached because the reference message is missing.
- Application_Watchdog (S3)
  – Sets error level FDCAN_TTOST.EL to 11 when the application failed to serve the application watchdog.The application watchdog is configured via FDCAN_TTOCF.AWL. It is served by reading register FDCAN_TTOST. When the watchdog is not served in time, bit FDCAN_TTOST.AWE and interrupt flag

FDCAN_TTIR.AW are set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode (FDCAN_CCCR.MON set to 1).

- CAN_Bus_Off (S3)
  - Entering CAN_Bus_Off state sets error level FDCAN_TTOST.EL to 11. CAN_Bus_Off state is signaled by FDCAN_PSR.BO = 1 and FDCAN_CCCR.INIT = 1.
- Scheduling_Error_2 (S2)
  - Sets error level FDCAN_TTOST.EL to 10 if the MSC of one Tx_Trigger has reached 7. In addition, the interrupt flag FDCAN_TTIR.SE2 is set. The error level FDCAN_TTOST.EL is reset to 00 at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.
- Tx_Overflow (S2)
  - Sets error level FDCAN_TTOST.EL to 10 when the Tx count is equal to or higher than the expected number of Tx_T riggers FDCAN_TTMLM.ENTT and a Tx_Trigger event occurs. In addition, the interrupt flag FDCAN_TTIR.TXO is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is no more than FDCAN_TTMLM.ENTT at the start of a new matrix cycle.
- Scheduling_Error_1 (S1)
  - Sets error level FDCAN_TTOST.EL to 01 if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than two, or if one of the MSCs of an exclusive Rx_Trigger has reached seven. In addition, the interrupt flag FDCAN_TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the error level FDCAN_TTOST.EL is reset to 00.
- Tx_Underflow (S1)
  - Sets error level FDCAN_TTOST.EL to 01 when the Tx count is less than the expected number of Tx_Triggers FDCAN_TTMLM.ENTT at the start of a new matrix cycle. In addition, the interrupt flag FDCAN_TTIR.TXU is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is at least FDCAN_TTMLM.ENTT at the start of a new matrix cycle.

57.4.15 TTCAN message handling

Reference message

For potential time masters the identifier of the reference message is configured via FDCAN_TTRMC.RID. No dedicated Tx buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte for TTCAN level 1 (that is, the first four data bytes for TTCAN level 0 and the first four data bytes for TTCAN level 2) is provided by the FSE.

In case the reference message Payload select FDCAN_TTRMC.RMPS is set, the rest of the reference message payload (level 1: bytes 2-8, levels 0, 2: bytes 5-6) is taken from Tx buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 444. Number of data bytes transmitted with a reference message

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4

Table 444. Number of data bytes transmitted with a reference message (continued)

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
1	0	4	1	4
1	1	4 + MBO	1 + MBO	4 + MBO

To send additional payload with the reference message in level 1 a DLC > 1 must be configured, for level 0 and level 2 a DLC > 4 is required. In addition, the transmission request pending bit FDCAN_TXBRP.TRP0 of message buffer 0 must be set (see *Table 444*). In case bit FDCAN_TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the reference identifier FDCAN_TTRMC.RID is used.

Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see *Rx handler*).

The message status count MSC is part of the corresponding trigger memory element and must be initialized to 0 during configuration. It is updated while the TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has been completed. This must take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

Message transmission

For time-triggered message transmission the TTCAN supplies 32 dedicated Tx buffers (see *Transmit pause*). A Tx FIFO or Tx queue is not available when the FDCAN is configured for time-triggered operation (FDCAN_TTOCF.OM = 01 or 10).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program must update the data regularly and on time, synchronized to the cycle time. The user is responsible that no partially updated messages are transmitted. To assure this the user must proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has been completed by reading FDCAN_TXBTO
- Update the Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

Tx_Trigger_Continous

- Issue a cancellation request to reset the Tx buffer request pending bit
- Check whether the cancellation has finished by reading FDCAN_TXBCF
- Update Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

The message MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission cannot be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and cannot be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message may have been started within its transmit enable window but was not started because transmission was disabled (TTCAN in error level S2 or user has disabled this particular message).

The Tx buffers may be managed dynamically, that is, several messages with different identifiers may share the same Tx buffer element. In this case the user must ensure that no transmission request is pending for the Tx buffer element to be reconfigured by checking FDCAN_TXBRP.

If a Tx buffer with pending transmission request should be updated, the user must first issue a cancellation request and check whether the cancellation has been completed by reading FDCAN_TXBCF before it starts updating.

The Tx handler transfers a message from the message RAM to its intermediate output buffer at the trigger element, which becomes active immediately before the Tx_Trigger element (which defines the beginning of the transmit window). During and after the transfer time the transmit message may not be updated and its FDCAN_TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger must be large enough to guarantee that the Tx handler can read four words from the message RAM even at high RAM access load from other modules.

Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx buffer request pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx buffer request pending remains set. When the transmission was not successful due to disturbances, it is repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message must arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it is repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission is not started if the corresponding transmission request pending flag is reset by a successful Tx cancellation.

In single transmit windows, the Tx handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmission is attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or canceled by the user), the fourth request is ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window are started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This must be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx buffer needs to be updated.

57.4.16 TTCAN interrupt and error handling

The TT interrupt register FDCAN_TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT interrupt enable register FDCAN_TTIE. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the FDCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, for example by disturbances on the CAN bus, they are handled by the FDCAN protocol failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (levels 0, 2 only) that require a reaction by the application program. With a stop watch event triggered by a rising edge on pin fdcan_swt internal time values are captured. The trigger time mark interrupt notifies the application that a specific Time_Base_Trigger is reached. The register time mark interrupt signals that the time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) is equal to the time mark FDCAN_TTTMK.TM. It can also be used to finish a gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

57.4.17 Level 0

TTCAN level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. In this mode the FDCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of FDCAN_TTOCF.GEN is ignored. External event-synchronized operation is not available in level 0. A synchronized time base is maintained by the transmission of reference messages.

In level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (FDCAN_TTIR.TTMI) is set when the cycle time has reached FDCAN_TTOCF.IRTO ´ 0x200, it reminds the user to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (FDCAN_TTIR.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (FDCAN_TTIR.RTMI) are also possible.

The reference message is configured as for level 2 operation. Received reference messages are recognized by the identifier configured in register FDCAN_TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the user sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- FDCAN_TTRMC
- FDCAN_TTOCF except EVTP, AWL, GEN
- FDCAN_TTMLM except ENTT, TXEW
- FDCAN_TURCF

Level 0 operation is controlled via:

- FDCAN_TTOCN except NIG, TMG, FGP, GCS, TTMIE
- FDCAN_TTGTP
- FDCAN_TTTMK
- FDCAN_TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- FDCAN_TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time FDCAN_TTOCF.IRTO 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

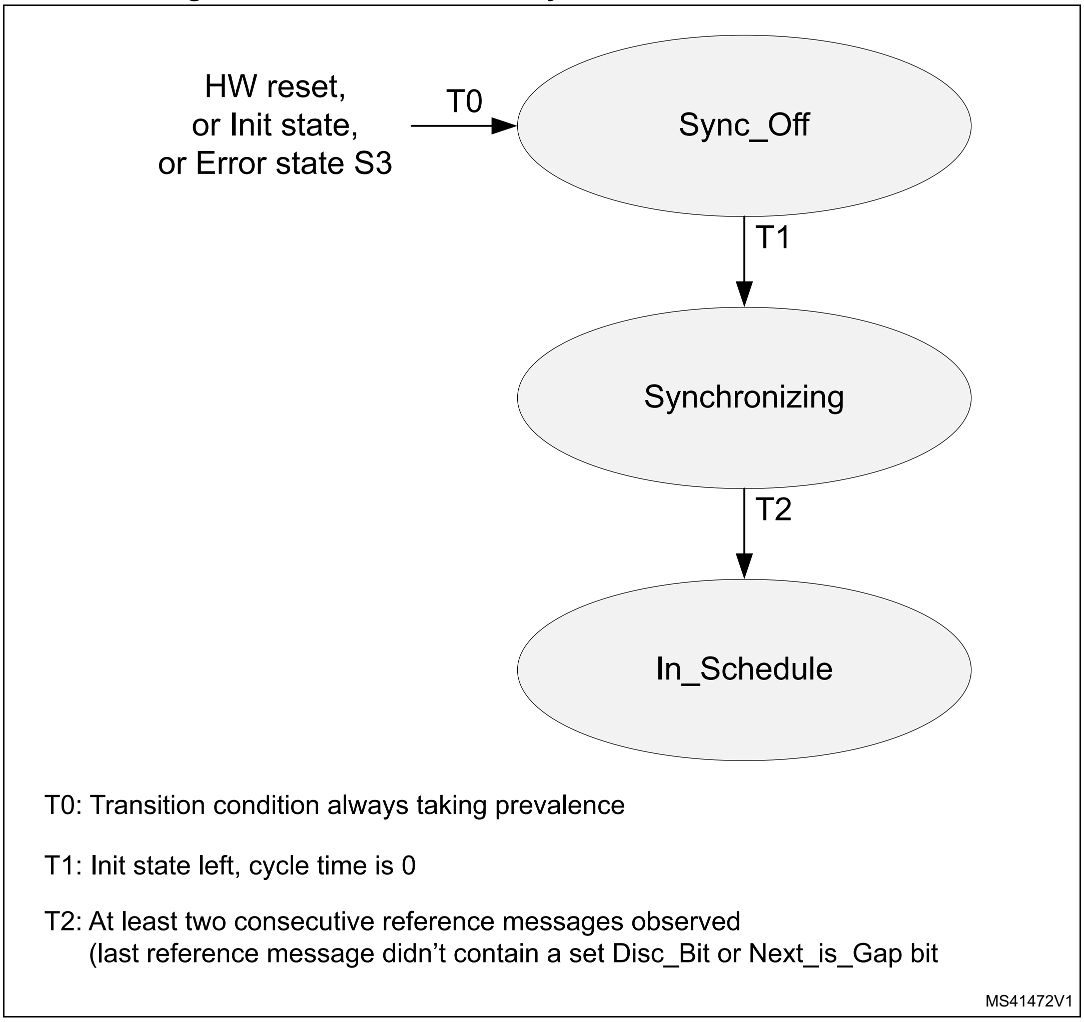
Level 0 operation is signaled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

Synchronizing

Figure 697 describes the states and the state transitions in TTCAN level 0 operation. level 0 has no In_Gap state.

Figure 697. Level 0 schedule synchronization state machine



Handling of error levels

During level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

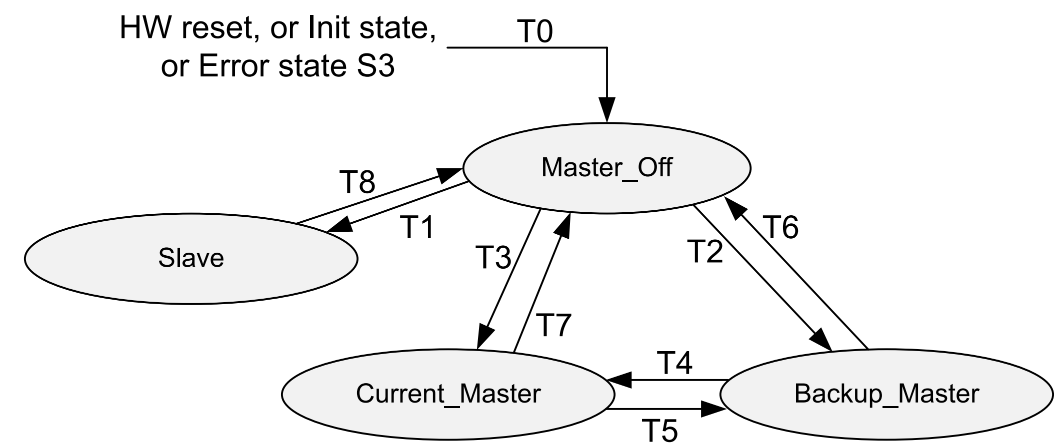
Since no S1 and S2 errors are possible, the error level can only switch between S0 (No error) and S3 (Severe error). In TTCAN level 0 an S3 error is handled differently. When error level S3 is reached, both FDCAN_TTOST.SYS and FDCAN_TTOST.MS are reset, and interrupt flags FDCAN_TTIR.GTE and FDCAN_TTIR.GTD are set.

When error level S3 (FDCAN_TTOST.EL = 11) is entered, bus monitoring mode is (contrary to TTCAN level 1 and level 2) not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

Master slave relation

Figure 698 describes the master slave relation in TTCAN level 0. In case of an S3 error the FDCAN returns to state Master_Off.

Figure 698. Level 0 master to slave relation



T0: Transition condition, always taking prevalence

T1: Reference message observed when not potential time master

T2: Reference message observed with master priority different from own master priority, Error state ≠ S3

T3: Reference message observed with master priority equal to own master priority, Error state ≠ S3

T4: Reference message observed with own master priority

T5: Reference message observed with master priority higher than own master priority

T6: Error state S3

T7: Error state S3

T8: Error state S3

MS41473V1

57.4.18 Synchronization to external time schedule

This feature can be used to synchronize the phase of the FDCAN schedule to an external schedule (for example that of a second TTCAN network or FlexRay network). It is applicable only when the FDCAN is current time master (FDCAN_TTOST.MS = 11).

External synchronization is controlled by event trigger input pin fdcan_evt. If bit FDCAN_TTOCN.ESCN is set, a rising edge at event trigger pin fdcan_evt the FDCAN compares its actual cycle time with the target phase value configured by FDCAN_TTGTP.CTP.

Before setting FDCAN_TTOCN.ESCN the user must adapt the phases of the two time schedules for example by using the FDCAN gap control (see *Section 57.4.11*). When the user sets FDCAN_TTOCN.ESCN, FDCAN_TTOSTSPL is set.

If the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is greater than 9 NTU, the phase lock bit FDCAN_TTOST.SPL is reset, and interrupt flag FDCAN_TTIR.CSM is set.

FDCAN_TTOST.SPL is also reset (and FDCAN_TTIR.CSM is set), when another node becomes time master.

If both FDCAN_TTOST.SPL and FDCAN_TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is lower or equal to nine NTU, the phase lock bit FDCAN_TTOST.SPL remains set, and the measured difference is used as the reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin fdcan_evt is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with FDCAN_*TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.*

57.4.19 FDCAN Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx buffer / FIFO element is shown in *Table 445*, the description is provided in *Table 446*.

Table 445. Rx buffer and FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="4">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>R0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="7">ID[28:0]</td></tr>
<tr><td>R1</td><td>ANMF</td><td colspan="3">FIDX[6:0]</td><td>Res.</td><td>FDF</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">RXTS[15:0]</td></tr>
<tr><td>R2</td><td colspan="4">DB3[7:0]</td><td colspan="4">DB2[7:0]</td><td>DB1[7:0]</td><td>DB0[7:0]</td></tr>
<tr><td>R3</td><td colspan="4">DB7[7:0]</td><td colspan="4">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>⋮</td><td colspan="4">⋮</td><td colspan="4">⋮</td><td colspan="2">⋮</td></tr>
<tr><td>Rn</td><td colspan="4">DBm[7:0]</td><td colspan="4">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_RXESC.

Table 446. Rx buffer and FIFO element description

Field	Description
R0 bit 31 ESI	Error state indicator 0: Transmitting node is error active 1: Transmitting node is error passive
R0 bit 30 XTD	Extended identifier Signals to the user whether the received frame has a standard or extended identifier. 0: 11-bit standard identifier 1: 29-bit extended identifier
R0 bit 29 RTR	Remote transmission request Signals to the user whether the received frame is a data frame or a remote frame. 0: Received frame is a data frame 1: Received frame is a remote frame

Table 446. Rx buffer and FIFO element description (continued)

Field	Description
R0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames may be enabled via FDCAN_GFC.ANFS and FDCAN_GFC.ANFE. 0: Received frame matching filter index FIDX 1: Received frame did not match any Rx filter element
R1 bits 30:24 FIDX[6:0]	Filter index 0-127 = index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to FDCAN_SIDFC.LSS. - 1 or FDCAN_XIDFC.LSE. - 1.
R1 bit 21 FDF	FD Format 0: Standard frame format 1: FDCAN frame format (new DLC-coding and CRC)
R1 bit 20 BRS	Bitrate Switch 0: Frame received without bitrate switching 1: Frame received with bitrate switching
R1 bits 19:16 DLC[3:0]	Data length code 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 15:0 RXTS[15:0]	Rx timestamp Timestamp counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.
R2 bits 31:24 DB3[7:0]	Data Byte 3
R2 bits 23:16 DB2[7:0]	Data Byte 2
R2 bits 15:8 DB1[7:0]	Data Byte 1
R2 bits 7:0 DB0[7:0]	Data Byte 0
R3 bits 31:24 DB7[7:0]	Data Byte 7
R3 bits 23:16 DB6[7:0]	Data Byte 6
R3 bits 15:8 DB5[7:0]	Data Byte 5
R3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮

Table 446. Rx buffer and FIFO element description (continued)

Field	Description
Rn bits 31:24 DBm[7:0]	Data Byte m
Rn bits 23:16 DBm-1[7:0]	Data Byte m-1
Rn bits 15:8 DBm-2[7:0]	Data Byte m-2
Rn bits 7:0 DBm-3[7:0]	Data Byte m-3

57.4.20 FDCAN Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx queue. The Tx handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx queue by evaluating the Tx buffer configuration FDCAN_TXBC.TFQS and FDCAN_TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_TXESC.

Table 447. Tx buffer and FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="5">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>T0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="8">ID[28:0]</td></tr>
<tr><td>T1</td><td colspan="4">MM[7:0]</td><td>EFC</td><td>Res.</td><td>FDF</td><td>BPS</td><td>DLC[3:0]</td><td colspan="2">Reserved</td></tr>
<tr><td>T2</td><td colspan="4">DB3[7:0]</td><td colspan="5">DB2[7:0]</td><td>DB1[7:0]</td><td>DB0[7:0]</td></tr>
<tr><td>T3</td><td colspan="4">DB7[7:0]</td><td colspan="5">DB6[7:0]</td><td>DB5[7:0]</td><td>DB4[7:0]</td></tr>
<tr><td>⋮</td><td colspan="4">⋮</td><td colspan="5">⋮</td><td colspan="2">⋮</td></tr>
<tr><td>Tn</td><td colspan="4">DBm[7:0]</td><td colspan="5">DBm-1[7:0]</td><td>DBm-2[7:0]</td><td>DBm-3[7:0]</td></tr>
</table>

Table 448. Tx buffer element description

Field	Description
T0 bit 31 ESI(1)	Error state indicator 0: ESI bit in CAN FD format depends only on error passive flag 1: ESI bit in CAN FD format transmitted recessive
T0 bit 30 XTD	Extended identifier 0: 11-bit standard identifier 1: 29-bit extended identifier
T0 bit 29 RTR(2)	Remote transmission request 0: Transmit data frame 1: Transmit remote frame

Table 448. Tx buffer element description (continued)

Field	Description
T0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
T1 bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 bit 23 EFC	Event FIFO control 0: Don't store Tx events 1: Store Tx events
T1 bit 21 FDF	FD Format 0: Frame transmitted in classic CAN format 1: Frame transmitted in CAN FD format
T1 bit 20 BRS(3)	Bitrate switching 0: CAN FD frames transmitted without bitrate switching 1: CAN FD frames transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 bits 31:24 DB3[7:0]	Data Byte 3
T2 bits 23:16 DB2[7:0]	Data Byte 2
T2 bits 15:8 DB1[7:0]	Data Byte 1
T2 bits 7:0 DB0[7:0]	Data Byte 0
T3 bits 31:24 DB7[7:0]	Data Byte 7
T3 bits 23:16 DB6[7:0]	Data Byte 6
T3 bits 15:8 DB5[7:0]	Data Byte 5
T3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮
Tn bits 31:24 DBm[7:0]	Data Byte m
Tn bits 23:16 DBm-1[7:0]	Data Byte m-1

Table 448. Tx buffer element description (continued)

Field	Description
Tn bits 15:8 DBm-2[7:0]	Data Byte m-2
Tn bits 7:0 DBm-3[7:0]	Data Byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node always transmit the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if FDCAN_CCCR.FDOE enables the transmission in CAN FD format.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled FDCAN_CCCR.FDOE = 1. Bit BRS is only evaluated when in addition FDCAN_CCCR.BRSE = 1.

57.4.21 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the user gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register FDCAN_TXEFS.

Table 449. Tx Event FIFO element

<table>
<tr><th>Bit</th><th colspan="4">31 24</th><th colspan="4">23 16</th><th>15 8</th><th>7 0</th></tr>
<tr><td>E0</td><td>ESI</td><td>XTD</td><td>RTR</td><td colspan="7">ID[28:0]</td></tr>
<tr><td>E1</td><td colspan="4">MM[7:0]</td><td>ET[1:0]</td><td>EDL</td><td>BRS</td><td>DLC[3:0]</td><td colspan="2">TXTS[15:0]</td></tr>
</table>

Table 450. Tx Event FIFO element description

Field	Description
E0 bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 bit 30 XTD	Extended Identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
E1 bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.

Table 450. Tx Event FIFO element description (continued)

Field	Description
E1 bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 bit 20 BRS	Bitrate switching – 0: Frame transmitted without bitrate switching – 1: Frame transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code – 0 - 8: Frame with 0-8 data bytes transmitted – 9 - 15: Frame with eight data bytes transmitted
E1 bits 15:0 TXTS[15:0]	Tx timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.

57.4.22 FDCAN standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FDCAN_SIDFC.FLSSA plus the index of the filter element (0 … 127).

Table 451. Standard message ID filter element

<table>
<tr><th>Bit</th><th colspan="3">31 24</th><th>23 16</th><th colspan="2">15 8</th><th>7 0</th></tr>
<tr><td>S0</td><td>SFT[1:0]</td><td>SFEC[2:0]</td><td colspan="2">SFID1[10:0]</td><td>Res.</td><td colspan="2">SFID2[10:0]</td></tr>
</table>

Table 452. Standard message ID filter element field description

Field		Description
Bit 31:30 SFT[1:0][(1)]		Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]		Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer or as debug message, configuration of FDCAN_SFT[1:0] ignored
Bits 26:16 SFID1[10:0]		Standard filter ID 1 First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 15:0	SFID2[15:10]	Standard filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: – SFEC = 001 ... 110 Second ID of standard ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	SFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one fdcan_pclk period in case the filter matches. SFID2[8] is used by the calibration unit.
	SFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

1. With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = “000”).

Note: In case a reserved value is configured, the filter element is considered disabled.

57.4.23 FDCAN extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended message ID filter element, its address is the filter list extended start address FDCAN_XIDFC.FLESA plus two times the index of the filter element (0 … 63).

Table 453. Extended message ID filter element

Bit	31	30	29	28 ... 0
F0	EFEC[2:0]			EFID1[28:0]
F1	EFT[1:0]		Reserved	EFID2[28:0]

Table 454. Extended message ID filter element field description

Field	Description
F0 bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer, configuration of EFT[1:0] ignored
F0 bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only FDCAN_XIDAM masking mechanism.
F1 bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), FDCAN_XIDAM mask not applied

Table 454. Extended message ID filter element field description (continued)

Field		Description
F1 bits 28:0	EFID2[10:0]	Extended filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: – SFEC = 001 ... 110 Second ID of extended ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	EFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one fdcan_pclk period in case the filter matches. EFID2[8] interface is used by the calibration unit.
	EFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

57.4.24 FDCAN trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a trigger memory element, its address is the trigger memory start address FDCAN_TTTMC.TMSA plus the index of the trigger memory element (0 … 63).

Table 455. Trigger memory element

<table>
<tr><th>Bit</th><th>31 24</th><th colspan="2">23 16</th><th>15 8</th><th colspan="4">7 0</th></tr>
<tr><td>T0</td><td colspan="3">TM[15:0]</td><td>Res. | CC[6:0]</td><td>Res.</td><td>TMIN</td><td>TMEX</td><td>TYPE[3:0]</td></tr>
<tr><td>T1</td><td>Res.</td><td>FTYPE</td><td>MNR[6:0]</td><td colspan="4">Res.</td><td>MSC[2:0]</td></tr>
</table>

Table 456. Trigger memory element description

Field	Description
T0 bits 31:16 TM[15:0]	Time mark Cycle time for which the trigger becomes active.
T0 bit 14:8 CC[6:0]	Cycle code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. – 0b000000x valid for all cycles – 0b000001c valid every 2nd cycle at cycle count mod2 = c – 0b00001cc valid every 4th cycle at cycle count mod4 = cc – 0b0001ccc valid every 8th cycle at cycle count mod8 = ccc – 0b001cccc valid every 16th cycle at cycle count mod16 = cccc – 0b01ccccc valid every 32nd cycle at cycle count mod32 = ccccc – 0b1cccccc valid every 64th cycle at cycle count mod64 = cccccc

Table 456. Trigger memory element description (continued)

Field	Description
T0 bit 5 TMIN	Time mark event internal – 0: No action – 1: FDCAN_TTIR.TTMI is set when trigger memory element becomes active
T0 bit 4 TMEX	Time mark event external – 0: No action – 1: Pulse at output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance with the length of one period is generated when the time arc of the trigger memory element becomes active and FDCAN_TTOCN.TTMIE = 1
T0 bit 3:0 TYPE[3:0]	Trigger type – 0000 Tx_Ref_Trigger - valid when not in gap – 0001 Tx_Ref_Trigger_Gap - valid when in gap – 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window – 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window – 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window – 0101 Tx_Trigger_Merged - starts a merged arbitration window – 0110 Watch_Trigger - valid when not in gap – 0111 Watch_Trigger_Gap - valid when in gap – 1000 Rx_Trigger - check for reception – 1001 Time_Base_Trigger - only control TMIN, TMEX – 1010 … 1111=End_of_List - illegal type, causes configuration error
T1 bit 23FTYPE	Filter type – 0: 11-bit standard message ID – 1: 29-bit extended message ID
T1 bit 22:16 MNR[6:0](1)	Message number – Transmission: trigger is valid for configured Tx buffer number. Valid values are 0 to 31. – Reception: trigger is valid for standard/extended message ID filter element number. Valid values are, respectively 0 to 63 and 0 to 127.
T1 bits 2:0 MSC[2:0]	Message status count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). – 0-7= Actual status

1. The trigger memory elements must be written when the FDCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual reference trigger offset (TTOST[RTO]).

57.5 FDCAN registers

57.5.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 3

Bits 27:24 **STEP[3:0]**: Step of core release = 2

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp year = 4

Bits 15:8 **MON[7:0]**: Timestamp month = 12

Bits 7:0 **DAY[7:0]**: Timestamp day =18

57.5.2 FDCAN Endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value
The endianness test value is 0x8765 4321.

57.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is dedicated to data bit timing phase and only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN time quantum may be programmed in the range from 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock periods.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (DTSEG1 + DTSEG2 + 3) tq for programmed values, or (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq for functional values.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
- 0: Transceiver delay compensation disabled
- 1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bitrate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of these quanta. Valid values for the baudrate prescaler are 0 to 31. The hardware interpreters this value as the programmed value plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is, t_{BS1} = (DTSEG1 + 1) x tq.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, that is, t_{BS2} = (DTSEG2 + 1) x tq.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, that is, t_{SJW} = (DSJW + 1) x tq.

Note: With an FDCAN clock of 8 MHz, the reset value 0x00000A33 configures the FDCAN for a fast bitrate of 500 kbit/s.

Note: The data phase bit rate must be higher than or equal to the nominal bit rate.

57.5.4 FDCAN test register (FDCAN_TEST)

Write access to this register must be enabled by setting bit FDCAN_CCCR.TEST to 1. All register functions are set to their reset values when bit FDCAN_CCCR.TEST is reset.

Loop back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

Monitors the actual value of transmit pin FDCANx_RX. This bit is a synchronized version of the FDCANx_RX, it goes back to 1 after reset if connected to a network.
0: The CAN bus is dominant (FDCANx_RX = 0)
1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time
01: Sample point can be monitored at pin FDCANx_TX
10: Dominant (0) level at pin FDCANx_TX
11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop back mode

0: Reset value, loop back mode is disabled
1: Loop back mode is enabled (see *Test modes*)

Bits 3:0 Reserved, must be kept at reset value.

57.5.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured by the FDCAN_RWD.WDC bits.

The counter is reloaded with FDCAN_RWD.WDC bits when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops and the interrupt flag FDCAN_IR.WDI bit is set. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00 the counter is disabled.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.5.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and resetting of single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP**:
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bitrate switching
0: Bitrate switching for transmissions disabled
1: Bitrate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable

0: Normal operation, register TEST holds reset values

1: Test mode, write access to register TEST enabled

Bit 6 **DAR**: Disable automatic retransmission

0: Automatic retransmission of messages not transmitted successfully enabled

1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode

Bit MON can be set only by software when both CCE and INIT are set to 1. The bit can be reset by the user at any time.

0: Bus monitoring mode is disabled

1: Bus monitoring mode is enabled

Bit 4 **CSR**: Clock stop request

0: No clock stop is requested

1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA**: Clock stop acknowledge

0: No clock stop acknowledged

1: FDCAN may be set in power down by stopping APB clock and kernel clock

Bit 2 **ASM**: ASM restricted operation mode

The restricted operation mode is intended for applications that adapt themselves to different CAN bitrates. The application tests different bitrates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can be set only by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time. This bit is set automatically set to 1 when the Tx handler was not able to read data from the message RAM in time. If the FDCAN is connected to a clock calibration on CAN unit, ASM bit is set by hardware as long as the calibration is not completed.

0: Normal CAN operation

1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable

0: The CPU has no write access to the protected configuration registers

1: The CPU has write access to the protected configuration registers (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Bit 0 **INIT**: Initialization

0: Normal operation

1: Initialization is started (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer must ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

57.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is dedicated to the nominal bit timing used during the arbitration phase, and is only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of [1 … 1024] FDCAN kernel clock periods.

tq = (BRP + 1) FDCAN clock period fdcan_tq_ck

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Should be smaller than NTSEG2, valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, that is, t_{SJW} = (NSJW + 1) x tq.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 24:16 **NBRP[8:0]**: Bitrate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The value used by the hardware is the one programmed, incremented by 1.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The value used by the hardware is the one programmed, incremented by 1 (t_{BS1} = (NTSEG1 + 1) x tq).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1 (t_{BS2} = (NTSEG2 + 1) x tq).

Note: With a CAN kernel clock of 8 MHz, the reset value of 0x00000A33 configures the FDCAN for a bitrate of 125 kbit/s.

57.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 … 16].
The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000
01: Timestamp counter value incremented according to TCP
10: External timestamp counter from TIM3 value used (tim3_cnt[0:15])
11: Same as 00.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.5.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When FDCAN_TSCC.TSS = 01, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. A wrap around sets interrupt flag FDCAN_IR.TSW. Write access resets the counter to 0. When FDCAN_TSCC.TSS = 10, TSC reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 not caused by write access to FDCAN_TSCV.

57.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in Continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **ETOC**: Enable timeout counter

0: Timeout counter disabled
1: Timeout counter enabled
This is a write-protected bit, write access is possible only when the bit 1 (CCE) and bit 0 (INIT) of FDCAN_CCCR register are set to 1.

For more details see *Timeout counter*.

57.5.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. When decremented to 0, interrupt flag FDCAN_IR.TOO is set and the timeout counter is stopped. Start and reset/restart conditions are configured via FDCAN_TOCC.TOS.

57.5.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag FDCAN_IR.ELO.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128

1: The receive error counter has reached the error passive level of 128

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When FDCAN_CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

57.5.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs_r	rs_r	rs_r	r	r	r	r	r	rs_r	rs_r	rs_r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and FDCAN_TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was reset by the CPU, no FDCAN message has been received

1: Message in FDCAN format with EDL flag set has been received

Access type is RX: reset on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set

1: Last received FDCAN message had its BRS flag set

Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not ha ve its ESI flag set

1: Last received FDCAN message had its ESI flag set

Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of an FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field is cleared to 0 when an FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status

0: The FDCAN is not Bus_Off

1: The FDCAN is in Bus_Off state

Bit 6 **EW**: Warning status

0: Both error counters are below the Error_Warning limit of 96

1: At least one of error counter has reached the Error_Warning limit of 96

Bit 5 **EP**: Error passive

0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected

1: The FDCAN is in the Error_Passive state

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication

01: Idle: node is neither receiver nor transmitter

10: Receiver: node is operating as receiver

11: Transmitter: node is operating as transmitter

Bits 2:0 **LEC[2:0]:** Last error code

The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error.

000: No error: No error occurred since LEC has been reset by successful reception or transmission.

001: Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error: A fixed format part of a received frame has the wrong format.

011: AckError: The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: NoChange: Any read access to the protocol status register re-initializes the LEC to ‘7. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is RS: set on read.

Note: When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) is shown in FLEC instead of LEC. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a Form error, not Stuff error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting FDCAN_CCCR.INIT. If the device goes Bus_Off, it sets FDCAN_CCCR.INIT of its own, stopping all bus activities. Once FDCAN_CCCR.INIT has been cleared by the CPU, the device waits for 129 occurrences of bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the error management counters are reset. During the waiting time after the reset of FDCAN_CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 error code is written to FDCAN_PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. FDCAN_ECR.REC is used to count these sequences.

57.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset
Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 mtq.
These are write-protected bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length
Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.
These are write-protected bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.5.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset clears the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 28 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)

Bit 27 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 26 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 25 **BO**: Bus_Off status
0: Bus_Off status unchanged
1: Bus_Off status changed

Bit 24 **EW**: Warning status
0: Error_Warning status unchanged
1: Error_Warning status changed

Bit 23 **EP**: Error passive
0: Error_Passive status unchanged
1: Error_Passive status changed

Bit 22 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow
1: Overflow of CAN error logging counter occurred

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRX**: Message stored to dedicated Rx buffer
The flag is set whenever a received message has been stored into a dedicated Rx buffer.
0: No Rx buffer updated
1: At least one received message stored into a Rx buffer

Bit 18 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 17 **MRAF**: Message RAM access failure
The flag is set when the Rx handler

- Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- Was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated or the New data flag for a dedicated Rx buffer is not set. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the user must reset FDCAN_CCCR.ASM.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 16 **TSW**: Timestamp wraparound
0: No timestamp counter wraparound
1: Timestamp counter wraparound

Bit 15 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0

Bit 14 **TEFF**: Tx event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bit 13 **TEFW**: Tx event FIFO watermark reached
0: Tx event FIFO fill level below watermark
1: Tx event FIFO fill level reached watermark

Bit 12 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element

Bit 11 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 10 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 9 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 8 **HPM**: High priority message
0: No high priority message received
1: High priority message received

Bit 7 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 6 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 5 **RF1W**: Rx FIFO 1 watermark reached
0: Rx FIFO 1 fill level below watermark
1: Rx FIFO 1 fill level reached watermark

Bit 4 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 3 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 2 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 1 **RF0W**: Rx FIFO 0 watermark reached
0: Rx FIFO 0 fill level below watermark
1: Rx FIFO 0 fill level reached watermark

Bit 0 **RF0N**: Rx FIFO 0 New message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

57.5.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register is signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res.	Res.	DRXE	TOOE	MRAFE	TSWE
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAE**: Access to Reserved address enable

Bit 28 **PEDE**: Protocol error in data phase enable

Bit 27 **PEAE**: Protocol error in Arbitration phase enable

Bit 26 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 25 **BOE**: Bus_Off status
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRXE**: Message stored to dedicated Rx buffer interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TEFWE**: Tx event FIFO watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **HPME**: High priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1WE**: Rx FIFO 1 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0WE**: Rx FIFO 0 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

57.5.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific interrupt flag from the interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line must be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.	Res.	DRXL	TOOL	MRAFL	TSWL
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAL**: Access to reserved address line

Bit 28 **PEDL**: Protocol error in data phase line

Bit 27 **PEAL**: Protocol error in arbitration phase line

Bit 26 **WDIL**: Watchdog interrupt line

Bit 25 **BOL**: Bus_Off status

Bit 24 **EWL**: Warning status interrupt line

Bit 23 **EPL**: Error passive interrupt line

Bit 22 **ELOL**: Error logging overflow interrupt line

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRXL**: Message stored to dedicated Rx buffer interrupt line

Bit 18 **TOOL**: Timeout occurred interrupt line

Bit 17 **MRAFL**: Message RAM access failure interrupt line

Bit 16 **TSWL**: Timestamp wraparound interrupt line

Bit 15 **TEFLL**: Tx event FIFO element Lost interrupt line

Bit 14 **TEFFL**: Tx event FIFO full interrupt line

Bit 13 **TEFWL**: Tx event FIFO watermark reached interrupt line

Bit 12 **TEFNL**: Tx event FIFO new entry interrupt line

Bit 11 **TFEL**: Tx FIFO empty interrupt line

Bit 10 **TCFL**: Transmission cancellation finished interrupt line

Bit 9 **TCL**: Transmission completed interrupt line

Bit 8 **HPML**: High priority message interrupt line

Bit 7 **RF1LL**: Rx FIFO 1 message lost interrupt line

Bit 6 **RF1FL**: Rx FIFO 1 full interrupt line

Bit 5 **RF1WL**: Rx FIFO 1 watermark reached interrupt line

Bit 4 **RF1NL**: Rx FIFO 1 new message interrupt line

Bit 3 **RF0LL**: Rx FIFO 0 message lost interrupt line

Bit 2 **RF0FL**: Rx FIFO 0 full interrupt line

Bit 1 **RF0WL**: Rx FIFO 0 watermark reached interrupt line

Bit 0 **RF0NL**: Rx FIFO 0 new message interrupt line

57.5.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1
- 0: Interrupt line fdcan_intr1_it disabled
- 1: Interrupt line fdcan_intr1_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

57.5.19 FDCAN global filter configuration register (FDCAN_GFC)

Global settings for message ID filtering. The global filter configuration register controls the filter path for standard and extended messages as described in *Figure 688* and *Figure 689*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept non-matching frames standard
Defines how received messages with 11-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard ID
1: Reject all remote frames with 11-bit standard ID
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard ID
1: Reject all remote frames with 29-bit standard ID
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.5.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC)

Settings for 11-bit standard message ID filtering.The standard ID filter configuration register controls the filter path for standard messages as described in *Figure 688*.

Address offset: 0x0084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLSSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSS[7:0]**: List size standard
0: No standard message ID filter
1-128: Number of standard message ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLSSA[13:0]**: Filter list standard start address
Start address of standard message ID filter list (32-bit word address, see *Table 451*).These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

57.5.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC)

Settings for 29-bit extended message ID filtering. The FDCAN extended ID filter configuration register controls the filter path for standard messages as described in *Figure 689*.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLESA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSE[7:0]**: List size extended
0: No extended message ID filter
1-128: Number of extended ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLESA[13:0]**: Filter list extended start address
Start address of extended message ID filter list (32-bit word address, see *Table 453: Extended message ID filter element*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

57.5.22 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0090

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID Mask
For acceptance filtering of extended frames the extended ID AND Mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to 1 the mask is not active.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.5.23 FDCAN high priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLST	FIDX[6:0]							MSI[1:0]		BIDX[5:0]					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list

Indicates the filter list of the matching filter element.

0: Standard filter list

1: Extended filter list

Bits 14:8 **FIDX[6:0]**: Filter index

Index of matching filter element. Range is 0 to FDCAN_SIDFC[LSS] - 1 or FDCAN_XIDFC[LSE] - 1.

Bits 7:6 **MSI[1:0]**: Message storage indicator

00: No FIFO selected

01: FIFO overrun

10: Message stored in FIFO 0

11: Message stored in FIFO 1

Bits 5:0 **BIDX[5:0]**: Buffer index

Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

57.5.24 FDCAN new data 1 register (FDCAN_NDAT1)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[31:0]**: New data[31:0]

The register holds the new data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

57.5.25 FDCAN new data 2 register (FDCAN_NDAT2)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[63:32]**: New data[63:32]

The register holds the new data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears clear the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

57.5.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)

Address offset: 0x00A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F0OM	F0WM[6:0]							Res.	F0S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F0OM**: FIFO 0 operation mode

FIFO 0 can be operated in blocking or in overwrite mode.

0: FIFO 0 blocking mode

1: FIFO 0 overwrite mode

Bits 30:24 **F0WM[6:0]**: FIFO 0 watermark
0: Watermark interrupt disabled
1-64: Level for Rx FIFO 0 watermark interrupt (FDCAN_IR.RF0W)
>64: Watermark interrupt disabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F0S[6:0]**: Rx FIFO 0 size
0: No Rx FIFO 0
1-64: Number of Rx FIFO 0 elements
>64: Values greater than 64 are interpreted as 64
The Rx FIFO 0 elements are indexed from 0 to F0S-1.

Bits 15:2 **F0SA[13:0]**: Rx FIFO 0 start address
Start address of Rx FIFO 0 in message RAM (32-bit word address, see *Figure 687*).

Bits 1:0 Reserved, must be kept at reset value.

57.5.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x00A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	F0PI[5:0]					
						rw	rw			rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of interrupt flag FDCAN_IR.RF0L. When FDCAN_IR.RF0L is reset, this bit is also reset.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F0PI[5:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F0GI[5:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F0FL**[6:0]: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0, range 0 to 64.

57.5.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x00A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F0AI[5:0]**: Rx FIFO 0 acknowledge index
After the user has read a message or a sequence of messages from Rx FIFO 0 it must write the buffer index of the last element read from Rx FIFO 0 to F0AI. This sets the Rx FIFO 0 get index FDCAN_RXF0S.F0GI to F0AI + 1 and update the FIFO 0 fill level FDCAN_RXF0S.F0FL.

57.5.29 FDCAN Rx buffer configuration register (FDCAN_RXBC)

Address offset: 0x00AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:2 **RBSA[13:0]**: Rx buffer start address
Configures the start address of the Rx buffers section in the message RAM (32-bit word address). Also used to reference debug messages A, B, C.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

57.5.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C)

Address offset: 0x00B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F1OM	F1WM[6:0]							Res.	F1S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F1SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F1OM**: FIFO 1 operation mode

FIFO 1 can be operated in blocking or in overwrite mode.
0: FIFO 1 blocking mode
1: FIFO 1 overwrite mode

Bits 30:24 **F1WM[6:0]**: Rx FIFO 1 watermark

0: Watermark interrupt disabled
1-64: Level for Rx FIFO 1 watermark interrupt (FDCAN_IR.RF1W)
>64: Watermark interrupt disabled.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F1S[6:0]**: Rx FIFO 1 size

0: No Rx FIFO 1
1-64: Number of Rx FIFO 1 elements
>64: Values greater than 64 are interpreted as 64
The Rx FIFO 1 elements are indexed from 0 to F1S - 1.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **F1SA[13:0]**: Rx FIFO 1 start address

start address of Rx FIFO 1 in message RAM (32-bit word address, see *Figure 687*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

57.5.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x00B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMS[1:0]		Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	F1PI[5:0]					
r	r					r	r			r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F1GI[5:0]						Res.	F1FL[6:0]						
		r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:30 **DMS[1:0]**: Debug message status
- 00: Idle state, wait for reception of debug messages
- 01: Debug message A received
- 10: Debug messages A, B received
- 11: Debug messages A, B, C received

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of interrupt flag FDCAN_IR.RF1L. When FDCAN_IR.RF1L is reset, this bit is also reset.
- 0: No Rx FIFO 1 message lost
- 1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0.

Bit 24 **F1F**: Rx FIFO 1 full
- 0: Rx FIFO 1 not full
- 1: Rx FIFO 1 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F1PI[5:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F1GI[5:0]**: Rx FIFO 1 get index

Rx FIFO 1 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F1FL[6:0]**: Rx FIFO 1 fill level

Number of elements stored in Rx FIFO 1, range 0 to 64

57.5.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x00B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F1AI[5:0]**: Rx FIFO 1 acknowledge index
After the user has read a message or a sequence of messages from Rx FIFO 1 it must write the buffer index of the last element read from Rx FIFO 1 to F1AI. This sets the Rx FIFO 1 get index FDCAN_RXF1S.F1GI. to F1AI + 1 and update the FIFO 1 fill level FDCAN_RXF1S.F1FL.

57.5.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC)

Configures the number of data bytes belonging to an Rx buffer / Rx FIFO element. Data field sizes higher than 8 bytes are intended for CAN FD operation only.

Address offset: 0x00BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	RBDS[2:0]			Res.	F1DS[2:0]			Res.	F0DS[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **RBDS[2:0]**: Rx buffer data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **F1DS[2:0]**: Rx FIFO 0 data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **F0DS[2:0]**: Rx FIFO 1 data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

57.5.34 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 29:24 **TFQS[5:0]**: Transmit FIFO/queue size
0: No Tx FIFO/queue
1-32: Number of Tx buffers used for Tx FIFO/queue
>32: Values greater than 32 are interpreted as 32.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **NDTB[5:0]**: Number of dedicated transmit buffers
0: No dedicated Tx buffers
1-32: Number of dedicated Tx buffers
>32: Values greater than 32 are interpreted as 32.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TBSA[13:0]**: Tx buffers start address
Start address of Tx buffers section in message RAM (32-bit word address, see *Figure 687*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

Note: The sum of TFQS and NDTB cannot be larger than 32. There is no check for erroneous configurations. The Tx buffers section in the message RAM starts with the dedicated Tx buffers.

57.5.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in register FDCAN_TXBRP. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]				
										r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0 Tx FIFO/queue not full
1 Tx FIFO/queue full

Bits 20:16 **TFQPI[4:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 31

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **TFGI[4:0]**:
Tx FIFO get index.
Tx FIFO read index pointer, range 0 to 31. Read as 0 when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TFFL[5:0]**: Tx FIFO free level

Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as 0 when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1).

Note: In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get index indicate the number of the Tx buffer starting with the first dedicated Tx buffers. For example: For a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

57.5.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC)

Configures the number of data bytes belonging to a Tx buffer element. Data field sizes >8 bytes are intended for CAN FD operation only.

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TBDS[2:0]**: Tx buffer data field size:

000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

57.5.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TRP[31:0]**:Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via register FDCAN_TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register FDCAN_TXBCR.

FDCAN_TXBRP bits are set only for those Tx buffers configured via FDCAN_TXBC. After an FDCAN_TXBRP bit has been set, a Tx scan (see *Filtering for Debug messages*) is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of register FDCAN_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding FDCAN_TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via FDCAN_TXBCF

after successful transmission together with the corresponding FDCAN_TXBTO bit

when the transmission has not yet been started at the point of cancellation

when the transmission has been aborted due to lost arbitration

when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately, the corresponding FDCAN_TXBRP bit is reset.

57.5.38 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AR[31:0]**:Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the user to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. FDCAN_TXBAR bits are set only for those Tx buffers configured via FDCAN_TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

57.5.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CR[31:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding cancellation request bit; writing a 0 has no impact.

This enables the user to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. FDCAN_TXBCR bits are set only for those Tx buffers configured via FDCAN_TXBC. The bits remain set until the corresponding FDCAN_TXBRP bit is reset.

0: No cancellation pending

1: Cancellation pending

57.5.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TO[31:0]**: Transmission occurred

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

57.5.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CF[31:0]**: Cancellation finished

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

57.5.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TIE[31:0]**: Transmission interrupt enable

Each Tx buffer has its own transmission interrupt enable bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

57.5.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFIE[31:0]**: Cancellation finished interrupt enable
Each Tx buffer has its own cancellation finished interrupt enable bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

57.5.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)

Address offset: 0x00F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EFSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **EFWM[5:0]**: Event FIFO watermark
0: Watermark interrupt disabled
1-32: Level for Tx event FIFO watermark interrupt (FDCAN_IR.TEFW)
>32: Watermark interrupt disabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **EFS[5:0]**: Event FIFO size.
0: Tx event FIFO disabled
1-32: Number of Tx event FIFO elements
>32: Values greater than 32 are interpreted as 32
The Tx event FIFO elements are indexed from 0 to EFS-1.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **EFSA[13:0]**: Event FIFO start address

Start address of Tx event FIFO in message RAM (32-bit word address, see *Figure 687*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

57.5.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	EFPI[4:0]				
						r	r				r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost

This bit is a copy of interrupt flag FDCAN_IR.TEFL. When FDCAN_IR.TEFL is reset, this bit is also reset.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full

0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **EFPI[4:0]**: Event FIFO put index

Tx event FIFO write index pointer, range 0 to 31.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **EFGI[4:0]**: Event FIFO get index

Tx event FIFO read index pointer, range 0 to 31.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **EFFL[5:0]**: Event FIFO fill level

Number of elements stored in Tx event FIFO, range 0 to 31.

57.5.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **EFAI[4:0]**: Event FIFO acknowledge index

After the user has read an element or a sequence of elements from the Tx event FIFO, it must write the index of the last element read from Tx event FIFO to EFAI. This sets the Tx event FIFO get index FDCAN_TXEFS.EFGI to EFAI + 1 and update the FIFO 0 fill level FDCAN_TXEFS.EFFL.

57.5.47 FDCAN register map

Table 457. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2 [3:0]				DSJW[3:0]			
	Reset value									0			0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX [1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
	Reset value																	0	0	0	0			0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0		0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS [1:0]	
	Reset value													0	0	0	0															0	0

Table 457. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS [1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1														0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003F	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRS	RESI	DLEC [2:0]			BO	EW	EP	ACT [1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	FDCAN_IR	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
	Reset value			0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res.	Res.	DRXE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
	Reset value			0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.	Res.	DRXL	TOOL	MRAFL	TSWL	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x006-0x007F	Reserved	Reserved																															
0x0080	FDCAN_GFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS [1:0]		ANFE [1:0]		RRFS	RRFE
	Reset value																											0	0	0	0	0	0
0x0084	FDCAN_SIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]								FLSSA[13:0]														Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0088	FDCAN_XIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]								FLESA[13:0]														Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 457. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0090	FDCAN _XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0094	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	FIDX[6:0]							MISI [1:0]		BIDX[5:0]					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	FDCAN _NDAT1	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x009C	FDCAN _NDAT2	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	FDCAN _RXF0C	F0OM	F0WM[6:0]							Res.	F0S[6:0]							F0SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00A4	FDCAN _RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0	F0F	Res.	Res.	F0PI[5:0]						Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
	Reset value							0	0			0	0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0
0x00A8	FDCAN _RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
	Reset value																											0	0	0	0	0	0
0x00AC	FDCAN_RXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBSA[13:0]														Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00B0	FDCAN _RXF1C	F1OM	F1WM[6:0]							Res.	F1S[6:0]							F1SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00B4	FDCAN _RXF1S	DMS [1:0]		Res.	Res.	Res.	Res.	RF1	F1F	Res.	Res.	F1PI[5:0]						Res.	Res.	F1GI[5:0]						Res.	F1FL[6:0]						
	Reset value	0	0					0	0			0	0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0
0x00B8	FDCAN _RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
	Reset value																											0	0	0	0	0	0
0x00BC	FDCAN _RXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBDS [2:0]				Res.	F1DS [2:0]				Res.	F0DS [2:0]		
	Reset value																				0	0	0			0	0	0			0	0	0
0x00C0	FDCAN_TXBC	Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]						TBSA[13:0]														Res.	Res.
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00C4	FDCAN _TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]					Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
	Reset value											0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x00C8	FDCAN _TXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS [2:0]		
	Reset value																														0	0	0

Table 457. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00CC	FDCAN_TXBRP	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D0	FDCAN_TXBAR	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D4	FDCAN_TXBCR	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	FDCAN_TXBTO	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	FDCAN_TXBCF	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E0	FDCAN_TXBTIE	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E4	FDCAN_TXBCIE	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00F0	FDCAN_TXEFC	Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]						EFSA[15:2]														Res.	Res.
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00F4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEF	EFF	Res.	Res.	Res.	EFPI[4:0]					Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
	Reset value							0	0				0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x00F8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
	Reset value																												0	0	0	0	0

Refer to *Section 2.5* for the register boundary addresses.

57.6 TTCAN registers

These registers are available only for FDCAN1.

57.6.1 FDCAN TT trigger memory configuration register (FDCAN_TTTMC)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TME[6:0]**: Trigger memory elements
0: No trigger memory
1-64: Number of trigger memory elements
>64: Values greater than 64 are interpreted as 64
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TMSA[13:0]**: Trigger memory start address.
Start address of trigger memory in message RAM (32-bit word address, see *Figure 687*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

57.6.2 FDCAN TT reference message configuration register (FDCAN_TTRMC)

Address offset: 0x0104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMPS	XTD	Res.	RID[28:16]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RMPS**: Reference message payload select
Ignored in case of time slaves.
0: Reference message has no additional payload
1: The following elements are taken from Tx buffer 0:
Message marker MM,
Event FIFO control EFC,
Data length code DLC,
Data Bytes DB (level 1: bytes 2-8, levels 0, 2: bytes 5-8)
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 30 **XTD**: Extended identifier
0: 11-bit standard identifier
1: 29-bit extended identifier
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 29 Reserved, must be kept at reset value.

Bits 28:0 **RID[28:0]**: Reference identifier.
Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.6.3 FDCAN TT operation configuration register (FDCAN_TTOCF)

Address offset: 0x0108

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECS	IRTO[6:0]							LDSDL[2:0]			TM	GEN	Res.	OM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **EVTP**: Event trigger polarity.
0: Rising edge trigger
1: Falling edge trigger
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 25 **ECC**: Enable clock calibration.
0: Automatic clock calibration in FDCAN levels 0, 2 is disabled
1: Automatic clock calibration in FDCAN levels 0, 2 is enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 24 **EGTF**: Enable global time filtering.
0: Global time filtering in FDCAN levels 0, 2 is disabled
1: Global time filtering in FDCAN levels 0, 2 is enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:16 **AWL[7:0]**: Application watchdog limit.
The application watchdog can be disabled by programming AWL to 0x00.
0x00 to FF: Maximum time after which the application must serve the application watchdog. The application watchdog is incremented once each 256 NTUs.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 15 **EECS**: Enable external clock synchronization
If enabled, TUR configuration (FDCAN_TURCF.NCL only) may be updated during FDCAN operation.
0: External clock synchronization in FDCAN levels 0, 2 disabled
1: External clock synchronization in FDCAN levels 0, 2 enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 14:8 **IRTO[6:0]**: Initial reference trigger offset.
0x00 to 7F Positive offset, range from 0 to 127
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:5 **LDSDL[2:0]**: LD of synchronization deviation limit.
The synchronization deviation limit SDL is configured by its dual logarithm LDSDL with SDL = 2 * (LDSDL + 5). SDL is comprised between 32 and 4096. It should not exceed the clock tolerance given by the CAN bit timing configuration.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 4 **TM**: Time master.
0: Time master function disabled
1: Potential time master
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 3 **GEN**: Gap enable.
0: Strictly time-triggered operation
1: External event-synchronized time-triggered operation
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **OM[1:0]**: Operation mode.
00: Event-driven CAN communication, default
01: TTCAN level 1
10: TTCAN level 2
11: TTCAN level 0
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

57.6.4 FDCAN TT matrix limits register (FDCAN_TTMLM)

Address offset: 0x010C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ENTT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TXEW[3:0]				CSS[1:0]		CCM[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ENTT[11:0]**: Expected number of Tx triggers
0x000 to FFF Expected number of Tx triggers in one matrix cycle.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TXEW[3:0]**: Tx enable window
0x0 to F Length of Tx enable window, 1-16 NTU cycles
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:6 **CSS[1:0]**: Cycle start synchronization
Enables sync pulse output.
00: No sync pulse
01: Sync pulse at start of basic cycle
10: Sync pulse at start of matrix cycle
11: Reserved
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 5:0 **CCM[5:0]:** Cycle count Max
0x00: 1 basic cycle per matrix cycle
0x01: 2 basic cycles per matrix cycle
0x03: 4 basic cycles per matrix cycle
0x07: 8 basic cycles per matrix cycle
0x0F: 16 basic cycles per matrix cycle
0x1F: 32 basic cycles per matrix cycle
0x3F: 64 basic cycles per matrix cycle
Others: Reserved
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: ISO 11898-4, Section 5.2.1 requires that only the listed cycle count values are configured. Other values are possible, but may lead to inconsistent matrix cycles.

57.6.5 FDCAN TUR configuration register (FDCAN_TURCF)

The length of the NTU is given by: NTU = CAN clock period x NC/DC.

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC extends from 0x10000 to 0x1FFFF. The value configured by NCL is the initial value for FDCAN_TURNA.NAV[15:0]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.

- Level 1: NC 4 × DC and NTU = CAN bit time
- Levels 0 and 2: NC 8 × DC

The actual value of FDCAN_TUR may be changed by the clock drift compensation function of TTCAN level 0 and level 2 in order to adjust the node local view of the NTU to the time master view of the NTU. DC is not changed by the automatic drift compensation, FDCAN_TURNA.NAV may be adjusted around NC in the range of the synchronization deviation limit given by FDCAN_TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in achieve the best computational accuracy for the drift compensation process.

Address offset: 0x0110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ELT	Res.	DC[13:0]													
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NCL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ELT**: Enable local time.

0: Local time is stopped, default

1: Local time is enabled

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: The local time is started by setting ELT, it remains active until ELT is reset or until the next hardware reset. FDCAN_TURCF.DC is locked when FDCAN_TURCF.ELT = 1. If ELT is written to 0, the readable value stays at 1 until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register is locked.

Bit 30 Reserved, must be kept at reset value.

Bits 29:16 **DC[13:0]**: Denominator configuration.

0x0000: Illegal value

0x0001 to 0x3FFF: Denominator configuration

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:0 **NCL[15:0]**: Numerator configuration low.

Write access to the TUR numerator configuration low is only possible during configuration with FDCAN_TURCF.ELT = 0 or if FDCAN_TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT configuration mode, the new value takes effect when FDCAN_TTOST.WECS is cleared to 0. NCL is locked FDCAN_TTOST.WECS is 1.

0x0000 to FFFF Numerator configuration low

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: If NC < 7 × DC in TTCAN level 1, subsequent time marks in the trigger memory must differ by at least two NTUs.

57.6.6 FDCAN TT operation control register (FDCAN_TTOCN)

Address offset: 0x0114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **LCKC**: TT operation control register locked.

Set by a write access to register FDCAN_TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.

0: Write access to FDCAN_TTOCN enabled

1: Write access to FDCAN_TTOCN locked

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ESCN**: External synchronization control

If enabled the FDCAN synchronizes its cycle time phase to an external event signaled by a rising edge at event trigger pin (see *Section 57.4.18: Synchronization to external time schedule*).

0: External synchronization disabled

1: External synchronization enabled

Bit 12 **NIG**: Next is gap.

This bit can be set only when the FDCAN is the actual time master and when it is configured for external event-synchronized time-triggered operation (FDCAN_TTOCF.GEN = 1)

0: No action, reset by reception of any reference message

1: Transmit next reference message with Next_is_Gap = 1

Bit 11 **TMG**: Time mark gap.

0: Reset by each reference message

1: Next reference message started when register time mark interrupt FDCAN_TTIR.RTMI is activated

Bit 10 **FGP**: Finish gap.
Set by the CPU, reset by each reference message
0: No reference message requested
1: Application requested start of reference message

Bit 9 **GCS**: Gap control select
0: Gap control independent from event trigger
1: Gap control by input event trigger pin

Bit 8 **TTIE**: Trigger time mark interrupt pulse enable
External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Schedule or In_Gap.
0: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance disabled
1: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance enabled

Bits 7:6 **TMC[1:0]**: Register time mark compare.
00: No register time mark interrupt generated
01: Register time mark interrupt if time mark = cycle time
10: Register time mark interrupt if time mark = local time
11: Register time mark interrupt if time mark = global time
Note: When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = 00, then reconfigure FDCAN_TTTMK, and finally set FDCAN_TMC to the intended time reference.

Bit 5 **RTIE**: Register time mark interrupt pulse enable.
Register time mark interrupts are configured by register FDCAN_TTTMK. A register time mark interrupt pulse with the length of one fdcan_tq_ck period is generated when time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) is equal to FDCAN_TTTMK.TM, independently from the synchronization state.
0: Register time mark interrupt output disabled
1: Register time mark interrupt output enabled

Bits 4:3 **SWS[1:0]**: Stop watch source.
00: Stop watch disabled
01: Actual value of cycle time is copied to FDCAN_TTCPT.SWV
10: Actual value of local time is copied to FDCAN_TTCPT.SWV
11: Actual value of global time is copied to FDCAN_TTCPT.SWV

Bit 2 **SWP**: Stop watch polarity.
0: Rising edge trigger
1: Falling edge trigger

Bit 1 **ECS**: External clock synchronization.
Writing a 1 to ECS sets FDCAN_TTOST.WECS if the node is the actual time master. ECS is reset after one APB clock period. The external clock synchronization takes effect at the start of the next basic cycle.

Bit 0 **SGT**: Set global time.
Writing a 1 to SGT sets FDCAN_TTOST.WGDT if the node is the actual time master. SGT is reset after one APB clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to FDCAN_TTGTP.

57.6.7 FDCAN TT global time preset register (FDCAN_TTGTP)

If TTOST.WGDT is set, the next reference message is transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = 1, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = 1 becomes valid or if the node is not the current time master. TP is locked while FDCAN_TTOST.WGTD = 1 after setting FDCAN_TTOCN.SGT until the reference message with Disc_Bit = 1 becomes valid or until the node is no longer the current time master.

Address offset: 0x0118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CTP[15:0]**: Cycle time target phase
CTP is write-protected while FDCAN_TTOCN.ESCN or FDCAN_TTOST.SPL are set (see *Section 57.4.18: Synchronization to external time schedule*).
0x0000 to FFFF defines the target value of cycle time when a rising edge of event trigger is expected

Bits 15:0 **TP[15:0]**: Time preset
TP is write-protected while FDCAN_TTOST.WGTD is set.
0x0000 to 7FFF next master reference mark = master reference mark + TP
0x8000 reserved
0x8001–FFFF Next master reference mark = master reference mark - (0x10000 - TP).

57.6.8 FDCAN TT time mark register (FDCAN_TTTMK)

A time mark interrupt (FDCAN_TTIR.TMI = 1) is generated when the time base indicated by FDCAN_TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

Address offset: 0x011C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]						
r									rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LCKM**: TT time mark register locked

Always set by a write access to registers FDCAN_TTOCN. Set by write access to register FDCAN_TTTMK when FDCAN_TTOCN.TMC 00. Reset when the registers have been synchronized into the CAN clock domain.

0: Write access to FDCAN_TTTMK enabled

1: Write access to FDCAN_TTTMK locked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **TICC[6:0]**: Time mark cycle code

Cycle count for which the time mark is valid.

0b000000x valid for all cycles

0b000001c valid every second cycle at cycle count mod2 = c

0b00001cc valid every fourth cycle at cycle count mod4 = cc

0b0001ccc valid every eighth cycle at cycle count mod8 = ccc

0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc

0b01ccccc valid every thirty-second cycle at cycle count mod32 = ccccc

0b1cccccc valid every sixty-fourth cycle at cycle count mod64 = cccccc

Bits 15:0 **TM[15:0]**: Time mark

0x0000 to FFFF time mark

Note: When using byte access to register FDCAN_TTTMK it is recommended to first disable the time mark compare function (FDCAN_TTOCN.TMC = 00) to avoid comparisons on inconsistent register values.

57.6.9 FDCAN TT interrupt register (FDCAN_TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register.

Address offset: 0x0120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT
													rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CER**: Configuration error

Trigger out of order.

0: No error found in trigger list

1: Error found in trigger list

Bit 17 **AW**: Application watchdog

0: Application watchdog served in time

1: Application watchdog not served in time

Bit 16 **WT**: Watch trigger
0: No missing reference message
1: Missing reference message (level 0: cycle time 0xFF00)

Bit 15 **IWTG**: Initialization watch trigger
The initialization is restarted by resetting IWT.
0 No missing reference message during system startup
1 No system startup due to missing reference message

Bit 14 **ELC**: Error level changed
Not set when error level changed during initialization.
0: No change in error level
1: Error level changed

Bit 13 **SE2**: Scheduling error 2
0: No scheduling error 2
1: Scheduling error 2 occurred

Bit 12 **SE1**: Scheduling error 1
0: No scheduling error 1
1: Scheduling error 1 occurred

Bit 11 **TXO**: Tx count overflow
0: Number of Tx trigger as expected
1: More Tx trigger than expected in one cycle

Bit 10 **TXU**: Tx count underflow
0: Number of Tx trigger as expected
1: Less Tx trigger than expected in one cycle

Bit 9 **GTE**: Global time error
Synchronization deviation SD exceeds limit specified by FDCAN_TTOCF.LDSDL, TTCAN levels 0, 2 only.
0: Synchronization deviation within limit
1: Synchronization deviation exceeded limit

Bit 8 **GTD**: Global time discontinuity
0: No discontinuity of global time
1: Discontinuity of global time

Bit 7 **GTW**: Global time wrap
0: No global time wrap occurred
1: Global time wrap from 0xFFFF to 0x0000 occurred

Bit 6 **SWE**: Stop watch event
0: No rising/falling edge at stop watch trigger pin detected
1 :Rising/falling edge at stop watch trigger pin detected

Bit 5 **TTMI:** Trigger time mark event internal
Internal time mark events are configured by trigger memory element TMIN (see *Section 57.4.24*).
Set when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Gap or In_Schedule.
0: Time mark not reached
1: Time mark reached (level 0: cycle time FDCAN_TTOCF.RTO x 0x200)

Bit 4 **RTMI**: Register time mark interrupt
Set when time referenced by TTOCN.TMC (cycle, local, or global) is equal to FDCAN_TTTMK.TM, independently from the synchronization state.
0: Time mark not reached
1: Time mark reached

Bit 3 **SOG**: Start of gap
0 No reference message seen with Next_is_Gap bit set
1 Reference message with Next_is_Gap bit set becomes valid

Bit 2 **CSM**: Change of synchronization mode
0: No change in master to slave relation or schedule synchronization
1: Master to slave relation or schedule synchronization changed,
also set when FDCAN_TTOST.SPL is reset

Bit 1 **SMC**: Start of matrix cycle
0: No matrix cycle started since bit has been reset
1: Matrix cycle started

Bit 0 **SBC**: Start of basic cycle
0: No basic cycle started since bit has been reset
1: Basic cycle started

57.6.10 FDCAN TT interrupt enable register (FDCAN_TTIE)

The settings in the TT interrupt enable register determine which status changes in the TT interrupt register result in an interrupt.

Address offset: 0x0124

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE
													rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERE**: Configuration error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 17 **AWE**: Application watchdog interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 16 **WTE**: Watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 15 **IWTE**: Initialization watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 14 **ELCE**: Change error level interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 13 **SE2E**: Scheduling error 2 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 12 **SE1E**: Scheduling error 1 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 11 **TXOE**: Tx count overflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 10 **TXUE**: Tx count underflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 9 **GTEE**: Global time error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 8 **GTDE**: Global time discontinuity interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 7 **GTWE**: Global time wrap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 6 **SWEE**: Stop watch event interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 5 **TTMIE:** Trigger time mark event internal interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 4 **RTMIE**: Register time mark interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 3 **SOGE**: Start of gap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 2 **CSME**: Change of synchronization mode interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 1 **SMCE**: Start of matrix cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 0 **SBCE**: Start of basic cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

57.6.11 FDCAN TT interrupt line select register (FDCAN_TTILS)

The TT interrupt line select register assigns an interrupt generated by a specific interrupt flag from the TT interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line must be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0128

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL
													rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERL**: Configuration error interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 17 **AWL**: Application watchdog interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 16 **WTL:** Watch trigger interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 15 **IWTL:** Initialization watch trigger interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 14 **ELCL:** Change error level interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 13 **SE2L:** Scheduling error 2 interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 12 **SE1L:** Scheduling error 1 interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 11 **TXOL:** Tx count overflow interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 10 **TXUL:** Tx count underflow interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 9 **GTEL:** Global time error interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 8 **GTDL:** Global time discontinuity interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 7 **GTWL:** Global time wrap interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 6 **SWEL:** Stop watch event interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 5 TTMIL: Trigger time mark event internal interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 4 **RTMIL:** Register time mark interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 3 **SOGL:** Start of gap interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 2 **CSML:** Change of synchronization mode interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 1 **SMCL:** Start of matrix cycle interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 0 **SBCL:** Start of basic cycle interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

57.6.12 FDCAN TT operation status register (FDCAN_TTOST)

Address offset: 0x012C

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPL	WECS	AWE	WFE	GSI	TMP[2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[7:0]								QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **SPL**: Schedule phase lock

The bit is valid only when external synchronization is enabled (FDCAN_TTOCN.ESCN = 1). In this case it signals that the difference between cycle time configured by FDCAN_TTGTP.CTP and the cycle time at the rising edge at event trigger pin is less than or equal to 9 NTU (see *Section 57.4.18: Synchronization to external time schedule*).

0: Phase outside range

1: Phase inside range

Bit 30 **WECS**: Wait for external clock synchronization.

0: No external clock synchronization pending

1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.

Bit 29 **AWE**: Application watchdog event

The application watchdog is served by reading FDCAN_TTOST. When the watchdog is not served in time, bit AWE is set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode.

0: Application watchdog served in time

1: Failed to serve application watchdog in time

Bit 28 **WFE**: Wait for event

0: No gap announced, reset by a reference message with Next_is_Gap = 0

1: Reference message with Next_is_Gap = 1 received

Bit 27 **GSI**: Gap started indicator

0: No gap in schedule, reset by each reference message and for all time slaves

1: Gap time after basic cycle has started

Bits 26:24 **TMP[2:0]**: Time master priority

0x0-7 Priority of actual time master

Bit 23 **GFI**: Gap finished indicator

Set when the CPU writes FDCAN_TTOCN.FGP, or by a time mark interrupt if TMG = 1, or via input pin (event trigger) if FDCAN_TTOCN.GCS = 1. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.

0: Reset at the end of each reference message

1: Gap finished by FDCAN

Bit 22 **WGTD**: Wait for global time discontinuity

0: No global time preset pending

1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = 1 or after it received a reference message.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:8 **RTO[7:0]**: Reference trigger offset

The reference trigger offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of-127 is reached. In case the FDCAN becomes time master (MS[1:0] = 11), the reset of RTO is delayed due to synchronization between user and CAN clock domain. For time slaves the value configured by FDCAN_TTOCF.IRTO is read.

0x00-FF Actual reference trigger offset value

Bit 7 **QCS**: Quality of clock speed

Only relevant in TTCAN level 0 and level 2, otherwise fixed to 1.

0: Local clock speed not synchronized to time master clock speed

1: Synchronization deviation ≤ SDL

Bit 6 **QGTP**: Quality of global time phase
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 0.
0: Global time not valid
1: Global time in phase with time master

Bits 5:4 **SYS[1:0]**: Synchronization state
00: Out of Synchronization
01: Synchronizing to FDCAN communication
10: Schedule suspended by gap (In_Gap)
11: Synchronized to schedule (In_Schedule)

Bits 3:2 **MS[1:0]**: Master state
00: Master_Off, no master properties relevant
01: Operating as time Slave
10: Operating as backup time master
11: Operating as current time master

Bits 1:0 **EL[1:0]**: Error level
00: Severity 0 - No error
01: Severity 1 - Warning
10: Severity 2 - error
11: Severity 3 - Severe error

57.6.13 FDCAN TUR numerator actual register (FDCAN_TURNA)

There is no drift compensation in TTCAN level 1 (NAV = NC). In TTCAN level 0 and level 2, the drift between the node local clock and the time master local clock is calculated. The drift is compensated when the synchronization deviation (difference between NC and the calculated NAV) is lower than 2 * (FDCAN_TTOCF.LDSDL + 5). With FDCAN_TTOCF.LDSDL < 7, this results in a maximum range for NAV of (NC - 0x1000) ≤ NAV ≤ (NC + 0x1000).

Address offset: 0x0130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:16]	
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **NAV[17:0]**: Numerator actual value
0x0EFFF: illegal value
0x0F000 to 20FFF: actual numerator value
0x21000: illegal value

57.6.14 FDCAN TT local and global time register (FDCAN_TTLGT)

Address offset: 0x0134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **GT[15:0]**: Global time

Non-fractional part of the sum of the node local time and its local offset (see *Section 57.4.13*).

0x0000 to FFFF Global time value of FDCAN network

Bits 15:0 **LT[15:0]**: Local time

Non-fractional part of local time, incremented once each local NTU (see *Section 57.4.13*).

0x0000 to FFFF Local time value of FDCAN node

57.6.15 FDCAN TT cycle time and count register (FDCAN_TTCTC)

Address offset: 0x0138

Reset value: 0x003F 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CC[5:0]**: Cycle count

0x00 to 3F Number of actual basic cycle in the system matrix

Bits 15:0 **CT[15:0]**: Cycle time

Non-fractional part of the difference of the node local time and Ref_Mark (see *Section 57.4.13*).

0x0000 to FFFF: cycle time value of FDCAN basic cycle

57.6.16 FDCAN TT capture time register (FDCAN_TTCPT)

Address offset: 0x013C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
										r	r	r	r	r	r

Bits 31:16 **SWV[15:0]**: Stop watch value

On a rising / falling edge (as configured via FDCAN_TTOCN.SWP) at the stop watch trigger pin, when FDCAN_TTOCN.SWS] is different from 00 and FDCAN_TTIR.SWE is 0, the actual time value as selected by FDCAN_TTOCN.SWS (cycle, local, global) is copied to SWV and TFDCAN_TIR.SWE is set to 1.Capturing of the next stop watch value is enabled by resetting FDCAN_TTIR.SWE.

0x0000 to FFFF Captured stop watch value

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CCV[5:0]**: Cycle count value

Cycle count value captured together with SWV.

0x00 to 3F: captured cycle count value

57.6.17 FDCAN TT cycle sync mark register (FDCAN_TTCSM)

Address offset: 0x0140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CSM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSM[15:0]**: Cycle sync mark

The cycle sync mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.

0x0000 to FFFF Captured cycle time

57.6.18 FDCAN TT trigger select register (FDCAN_TTTS)

The settings in the FDCAN_TTTS register select the input to be used as event trigger and stop watch trigger.

Address offset: 0x0300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL[1:0]		Res.	Res.	SWTDEL[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **EVTSEL[1:0]**: Event trigger input selection
These bits are used to select the input to be used as event trigger
00: fdcan1_evt0
01: fdcan1_evt1
10: fdcan1_evt2
11: fdcan1_evt3

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SWTDEL[1:0]**: Stop watch trigger input selection
These bits are used to select the input to be used as stop watch trigger
00: fdcan1_swt0
01: fdcan1_swt1
10: fdcan1_swt2
11: fdcan1_swt3

57.6.19 FDCAN TT register map

Table 458. FDCAN TT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0100	FDCAN _TTTMC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]							TMSA[13:0]														Res.	Res.
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0104	FDCAN _TTRMC	RMPS	XTD	Res.	RID[28:0]																												
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0108	FDCAN _TTOCF	Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]								EECS	IRTO[6:0]							LDSDL [2:0]			TM	GEN	Res.	OM [1:0]	
	Reset value						0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x010C	FDCAN _TTMLM	Res.	Res.	Res.	Res.	ENTT[11:0]												Res.	Res.	Res.	Res.	TXEW[3:0]				CSS [1:0]		CCM[5:0]					
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 458. FDCAN TT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0110	FDCAN_TURCF	ELT	Res.	DC[13:0]														NCL[15:0]															
	Reset value	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114	FDCAN_TTOCN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0118	FDCAN_TTGTP	CTP[15:0]																TP[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x011C	FDCAN_TTTMK	LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]							TM[15:0]															
	Reset value	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0120	FDCAN_TTIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT	IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124	FDCAN_TTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0128	FDCAN_TTILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x012C	FDCAN_TTOST	SPL	WECS	AWE	WFE	GSI	TMP[2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.	RTO[7:0]								QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x0130	FDCAN_TURNA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134	FDCAN_TTLGT	GT[15:0]																LT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0138	FDCAN_TTCTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]						CT[15:0]															
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x013C	FDCAN_TTCPT	SWV[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x0140	FDCAN_TTCSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0144 to 0x01FC	Reserved																																
	Reset value																																

Table 458. FDCAN TT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0300	FDCAN _TTTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL [1:0]		Res.	Res.	SWTSEL [1:0]	
	Reset value																											0	0			0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

57.7 CCU registers

57.7.1 Clock calibration unit core release register (FDCAN_CCU_CREL)

Address offset: 0x0000

Reset value: 0x1114 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 1

Bits 27:24 **STEP[3:0]**: Step of core release = 1

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp year = 4

Bits 15:8 **MON[7:0]**: Timestamp month = 12

Bits 7:0 **DAY[7:0]**: Timestamp day = 18

57.7.2 Calibration configuration register (FDCAN_CCU_CCFG)

Address offset: 0x0004

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]			
rw												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 **SWR**: Software reset

Writing a 1 to this bit resets the calibration FSM to state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). The calibration watchdog value CWD.WDV is also reset. Registers FDCAN_CCFG, FDCAN_CCU_CSTAT and the calibration watchdog configuration CWD.WDC are unchanged. The bit remains set until reset is completed.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:16 **CDIV[3:0]**: Clock divider

The clock divider must be configured when the clock calibration is bypassed (BCC = 1) to ensure that the FDCAN requirement is fulfilled.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bit FDCAN_CCCR.CCE = 1.

Bits 15:8 **OCPM[7:0]**: Oscillator clock periods minimum

Configures the minimum number of periods in two CAN bit times. OCPM is used in basic calibration to avoid false measurements in case of glitches on the bus line. The configured number of periods is OCPM × 32. The configuration depends on the frequency and the bitrate configured in FDCAN modules (from 125 kbit/s up to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz fdcan_ker_ck and 1 Mbit/s CAN bitrate.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 7 **CFL**: Calibration field length

0: Calibration field length is 32 bits
1: Calibration field length is 64 bits

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 6 **BCC**: Bypass clock calibration

If this bit is set, the clock input fdcan_ker_ck is routed to the time quanta clock through a clock divider configurable via CDIV. In this case the baudrate prescaler of the connected FDCANs must be configured to generate the FDCAN internal time quanta clock.

0: Clock calibration unit generates time quanta clock
1: Clock calibration unit bypassed (default configuration)

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **TQBT[4:0]**: Time quanta per bit time

Configures the number of time quanta per bit time. Same value as configured in FDCAN modules. The range of the resulting time quanta clock fdcan_tq_ck is from 0.5 MHz (bitrate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bitrate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

57.7.3 Calibration status register (FDCAN_CCU_CSTAT)

Address offset: 0x0008

Reset value: 0x0203 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALS[1:0]		Res.	TQC[10:0]											OCPC[17:16]	
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 **CALS[1:0]**: Calibration state

00: Not_Calibrated
01: Basic_Calibrated
10: Precision_Calibrated
11: Reserved

Bit 29 Reserved, must be kept at reset value.

Bits 28:18 **TQC[10:0]**: Time quanta counter

Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

Bits 17:0 **OCPC[17:0]**: Oscillator clock period counter

Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

57.7.4 Calibration watchdog register (FDCAN_CCU_CWD)

Address offset: 0x000C

Reset value: 0x0000 0000

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). In this state the calibration watchdog monitors the message received. In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM stays in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01), the calibration watchdog is restarted with each received message . In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM returns to state

Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message is received, state Precision_Calibrated (FDCAN_CCU_CSTAT.CALS = 10) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the quartz message received input. In case no message from a quartz controlled node is received by the attached TTCAN until the calibration watchdog has counted down to 0, the calibration FSM transits back to state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01). The signal is active when the CAN protocol engine on the attached TTCAN is started i.e. when the INIT bit is reset.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE, the interrupt line is activated (set to high). Interrupt line remains active until the interrupt flag FDCAN_CCU_IR.CWE is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WDV[15:0]**: Watchdog value

Actual calibration watchdog counter value.

Bits 15:0 **WDC[15:0]**: WDC

Watchdog configuration

Start value of the calibration watchdog counter. With the reset value of 00 the counter is disabled.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

57.7.5 Clock calibration unit interrupt register (FDCAN_CCU_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register. The configuration of FDCAN_CCU_IE controls whether an interrupt is generated or not.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSC**: Calibration state changed
0: Calibration state unchanged
1: Calibration state has changed

Bit 0 **CWE**: Calibration watchdog event
0: No calibration watchdog event
1: Calibration watchdog event occurred

57.7.6 Clock calibration unit interrupt enable register (FDCAN_CCU_IE)

Address offset: 0x0014

Reset value: 0x0000 0000

The settings in the CU interrupt enable register determine whether a status change in the CU interrupt register is signaled on an interrupt line.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSCE**: Calibration state changed enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **CWEE**: Calibration watchdog event enable
0: Interrupt disabled
1: Interrupt enabled

57.7.7 CCU register map

Table 459. CCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CCU_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_CCU_CCFG	SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]				OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
	Reset value	0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	FDCAN_CCU_CSTAT	CALS [1:0]		Res.	TQC[10:0]											OCPC[17:0]																	
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	FDCAN_CCU_CWD	WDV[15:0]																WDC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 459. CCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0010	FDCAN_ CCU_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
	Reset value																															0	0
0x0014	FDCAN_ CCU_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
	Reset value																															0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

58 USB on-the-go high-speed (OTG)

58.1 Introduction

This section presents the architecture and the programming model of the OTG controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
HS	High-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification.* OTG supports the speeds defined in the *Table 460: OTG speeds supported* below.The USB OTG supports both HNP and SRP. The only external device required is a charge pump for V_{BUS} in OTG mode.

Table 460. OTG speeds supported

-	HS (480 Mb/s)	FS (12 Mb/s)	LS (1.5 Mb/s)
Host mode	X	X	X
Device mode	X	X	-

58.2 OTG main features

The main features can be divided into three categories: general, host-mode and device-mode features.

58.2.1 General features

The OTG interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- OTG supports the following PHY interfaces:
  - An on-chip full-speed PHY
  - A UTMI interface for internal HS PHY
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - Integrated support for host Negotiation protocol (HNP) and session request protocol (SRP)
  - It allows host to turn V_{BUS} off to conserve battery power in OTG applications
  - It supports OTG monitoring of V_{BUS} levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - SRP capable USB HS Peripheral (B-device)
  - SRP capable USB HS/LS host (A-device)
  - USB On-The-Go Full-Speed Dual Role device
- It supports HS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- OTG embeds an internal DMA with thresholding support and software selectable AHB burst type in DMA mode.
- It supports Descriptor-Based Scatter/Gather DMA controller for device and host mode. (Descriptor-Based Congruent-Sequential DMA is not supported). Scatter/Gather DMA operation is supported in both device and host mode. This feature will improve performance for device mode isochronous endpoints. Note that hubs (split transfers)

are not supported in host scatter/gather DMA mode of operation. Split transfers are supported only in host buffer DMA (internal DMA) mode of operation.

- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 4 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2 on the FS PHY transceiver only.

58.2.2 Host-mode features

The OTG interface main features and requirements in host-mode are the following:

- External charge pump for V_{BUS} voltage generation.
- Up to 16 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 16 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 16 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

58.2.3 Peripheral-mode features

The OTG interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 8 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 8 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 9 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

58.3 OTG implementation

Table 461. OTG implementation[1]

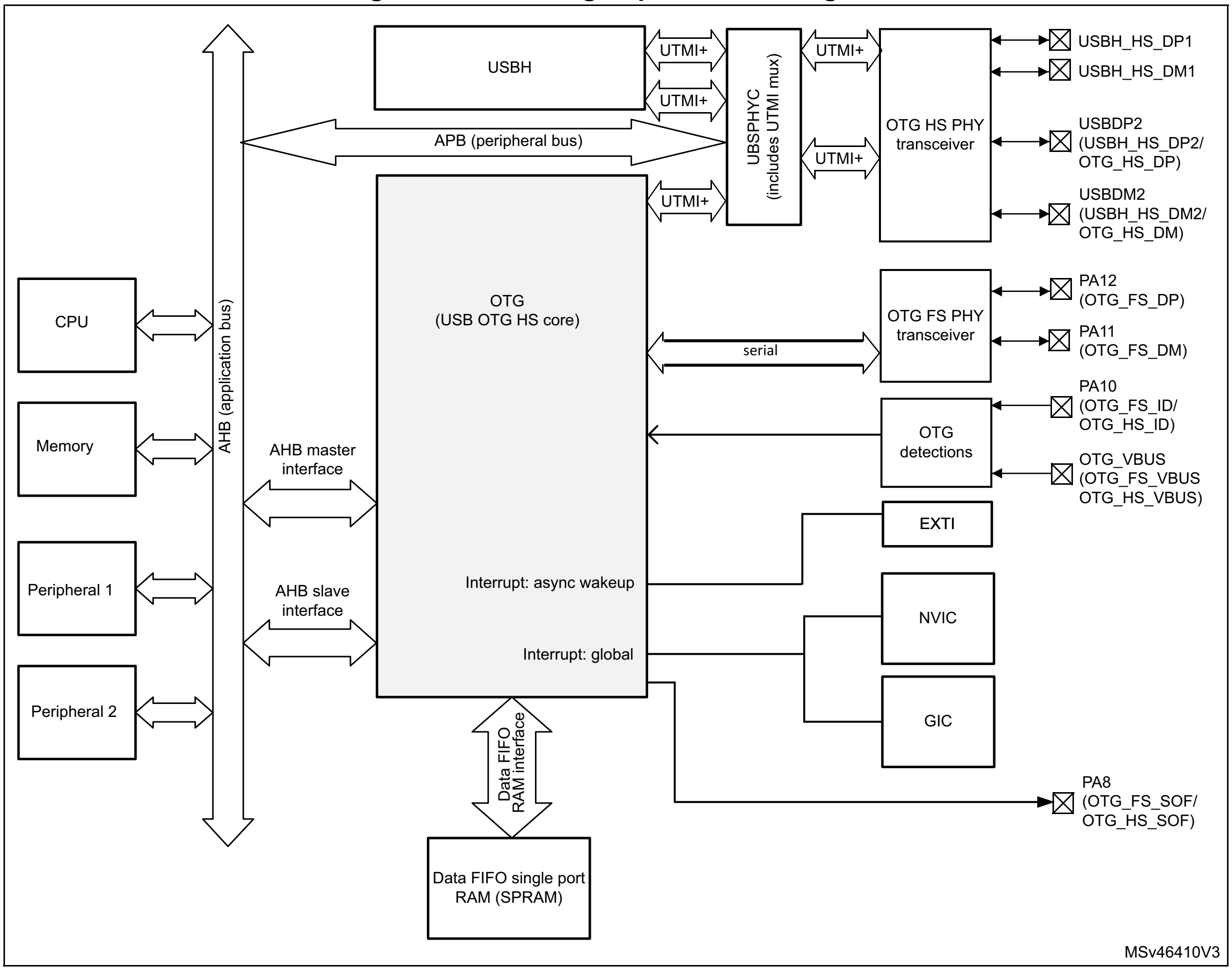
USB features	OTG for STM32MP153
Device bidirectional endpoints (including EP0)	9
Host mode channels	16
Size of dedicated SRAM	4 KB
USB 2.0 link power management (LPM) support	X
OTG revision supported	2.0
Attach detection protocol (ADP) support	-
Battery charging detection (BCD) support	X
ULPI available to primary IOs via muxing	-
Integrated PHY	FS and HS
Scatter/gather DMA	X
Dedicated interrupt lines for EP1	-

1. “X” = supported, “-” = not supported, “FS” = supported in FS mode, “HS” = supported in HS mode.

58.4 OTG functional description

58.4.1 OTG block diagram

Figure 699. OTG high-speed block diagram



58.4.2 OTG pin and internal signals

Table 462. OTG input/output pins

Signal name	Signal type	Description
OTG_HS_DP	Digital input/output	USB OTG D+ line
OTG_HS_DM	Digital input/output	USB OTG D- line
OTG_FS_DP	Digital input/output	USB OTG D+ line (FS PHY)
OTG_FS_DM	Digital input/output	USB OTG D- line (FS PHY)
OTG_HS_ID	Digital input	USB OTG ID
OTG_HS_VBUS	Analog input	USB OTG VBUS
OTG_HS_SOF	Digital output	USB OTG Start Of Frame (visibility)

Table 462. OTG input/output pins (continued)

Signal name	Signal type	Description
OTG_FS_ID	Digital input/output	USB OTG ID (use this pin with FS PHY)
OTG_FS_VBUS	Digital input/output	USB OTG VBUS (use this pin with FS PHY)
OTG_FS_SOF	Digital input/output	USB OTG Start Of Frame (visibility, use this pin with FS PHY)

Table 463. OTG input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wake-up event output
usb_gbl_it	Digital output	USB OTG global interrupt

58.4.3 OTG core

The USB OTG receives the 48 MHz clock from the reset and clock controller (RCC), see *Figure 81: Peripheral clock distribution for USB*. This clock is used for driving the 48 MHz domain at full-speed (12 Mbit/s) and must be enabled prior to configuring the OTG core.

The OTG receives the 60 MHz clock from the reset and clock controller (RCC). This is typically generated in the PLL associated with the HS PHY and enabled in the RCC. This clock is used for driving the 60 MHz domain at high-speed (480 Mbit/s) and must be enabled prior to configuring the OTG core.

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in *Section 58.12: OTG interrupts*.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 4-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG peripheral, the AHB frequency should be higher than 30 MHz.

58.4.4 Embedded full-speed OTG PHY connected to OTG

The embedded full-speed OTG PHY is controlled by the OTG core and conveys USB control & data signals through the full-speed subset of the UTMI+ Bus (UTMIFS). It provides

the physical support to USB connectivity.
The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as V_{BUS} is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the peripheral role is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of 2 resistors controlled separately from the OTG as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows to achieve a better noise rejection and Tx/Rx signal quality.

58.4.5 OTG detections

Additionally the OTG uses the following functions:

- integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- V_{BUS} sensing comparators with hysteresis used to detect V_{BUS} valid, A-B session valid and session-end voltage thresholds. They are used to drive the session request protocol (SRP), detect valid startup and end-of-session conditions, and constantly monitor the V_{BUS} supply during USB operations.

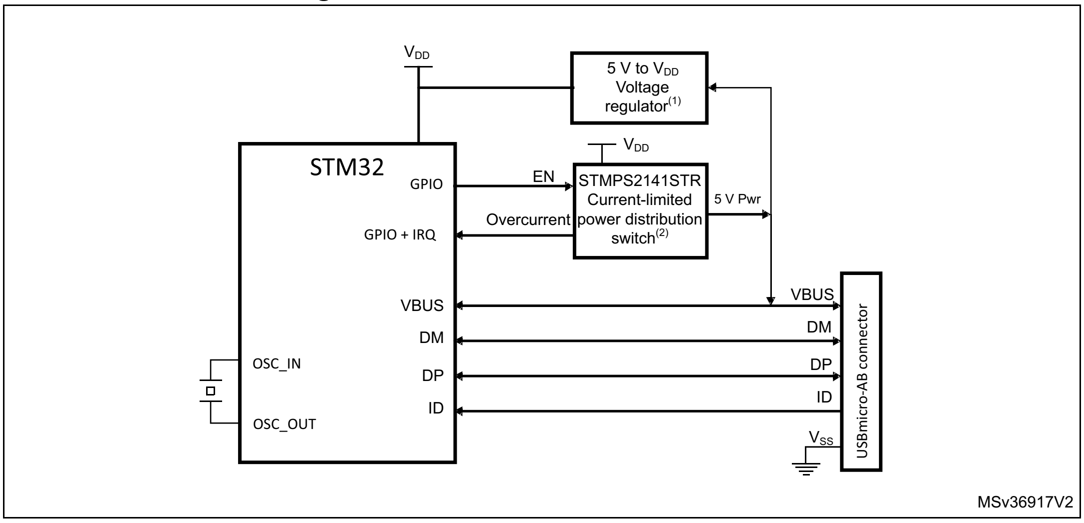
58.4.6 High-speed OTG PHY connected to OTG_HS

Note: Refer to implementation table to determine if an HS PHY is embedded.

The USB OTG core includes an internal UTMI interface which is connected to the embedded HS PHY (see *Section 58.4.1: OTG block diagram*).

58.5 OTG dual role device (DRD)

Figure 700. OTG A-B device connection



1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

58.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

58.5.2 HNP dual role device

The HNP capable bit in the Global USB configuration register (HNPCAP bit in OTG_GUSBCFG) enables the OTG core to dynamically change its role from A-host to A-peripheral and vice-versa, or from B-Peripheral to B-host and vice-versa according to the host negotiation protocol (HNP). The current device status can be read by the combined values of the connector ID status bit in the Global OTG control and status register (CIDSTS bit in OTG_GOTGCTL) and the current mode of operation bit in the global interrupt and status register (CMOD bit in OTG_GINTSTS).

The HNP program model is described in detail in *Section 58.15: OTG programming model*.

58.5.3 SRP dual role device

The SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG core to switch off the generation of V_{BUS} for the A-device to save power. Note that the A-device is always in charge of driving V_{BUS} regardless of the host or peripheral role of the OTG.

The SRP A/B-device program model is described in detail in *Section 58.15: OTG programming model*.

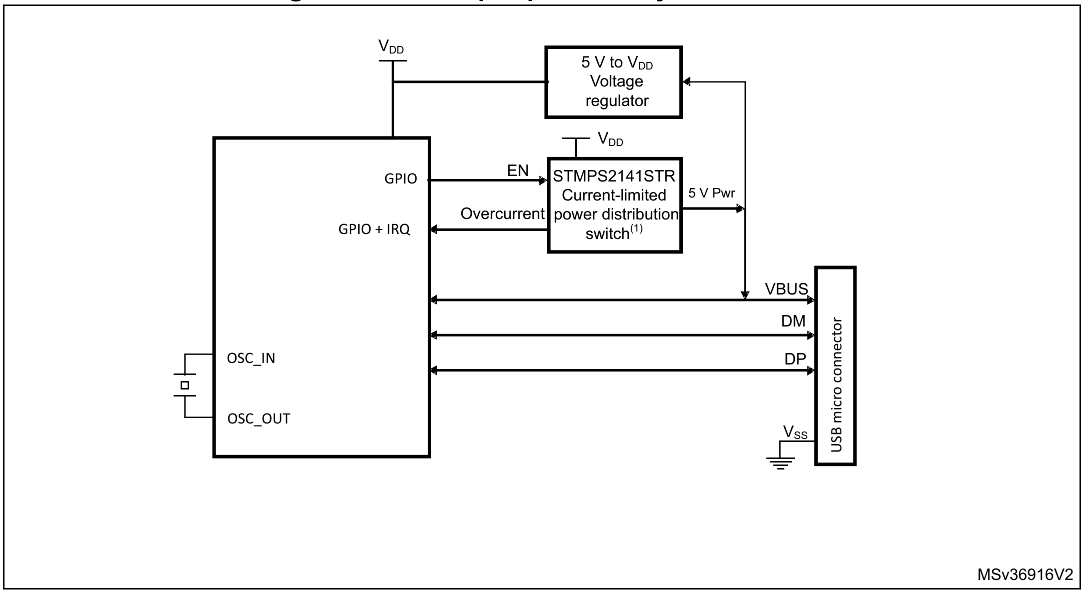
58.6 OTG as a USB peripheral

This section gives the functional description of the OTG in the USB peripheral mode. The OTG works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- OTG A-Peripheral
  - OTG A-device state after the HNP switches the OTG to its peripheral role
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable, and the HNP-capable bit in the Global USB Configuration register (HNPCAP bit in OTG_GUSBCFG) is cleared.
- Peripheral only
  - The force device mode bit (FDMOD) in the *Section 58.14.4: OTG USB configuration register (OTG_GUSBCFG)* is set to 1, forcing the OTG core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from V_{BUS}.

Figure 701. OTG peripheral-only connection



1. Use a regulator to build a bus-powered device.

58.6.1 SRP-capable peripheral

The SRP capable bit in the Global USB configuration register (SRPCAP bit in OTG_GUSBCFG) enables the OTG to support the session request protocol (SRP). In this way, it allows the remote A-device to save power by switching off V_{BUS} while the USB session is suspended.

The SRP peripheral mode program model is described in detail in the *B-device session request protocol* section.

58.6.2 Peripheral states

Powered state

The V_{BUS} input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The V_{BUS} input also ensures that valid V_{BUS} levels are supplied by the host during USB operations. If a drop in V_{BUS} below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG has exited the powered state.

In the powered state, the OTG expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wake-up signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

58.6.3 Peripheral endpoints

The OTG core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/)OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints
- 8 IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG_DIEPCTL*x*), transfer configuration (OTG_DIEPTSIZ*x*), and status-interrupt (OTG_DIEPINT*x*) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on

which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

- 8 OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG_DOEPCTL*x*), transfer configuration (OTG_DOEPTSIZ*x*) and status-interrupt (OTG_DOEPINT*x*) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

Endpoint control

- The following endpoint controls are available to the application through the device endpoint-*x* IN/OUT control register (OTG_DIEPCTL*x*/OTG_DOEPCTL*x*):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-*x* transfer size registers (OTG_DIEPTSIZ*x*/OTG_DOEPTSIZ*x*) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG core updates them with the current transfer status.

The following transfer parameters can be programmed:

- Transfer size in bytes
- Number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-*x* interrupt registers (OTG_DIEPINT*x*/OTG_DOPEPINT*x)* indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-*x* interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt

58.7 OTG as a USB host

This section gives the functional description of the OTG in the USB host mode. The OTG works as a USB host in the following circumstances:

Automatic host mode direct from ID pin:

The ID pin is not always available, refer to product datasheet for available pins.

- Use cases:
  - Micro connector: A-Device with a Micro-A plug inserted, and connector ID pin connected to the STM32 ID pin which will then be detected in a low state.
- Note that the integrated pulldown resistors are automatically set on the DP/DM lines.

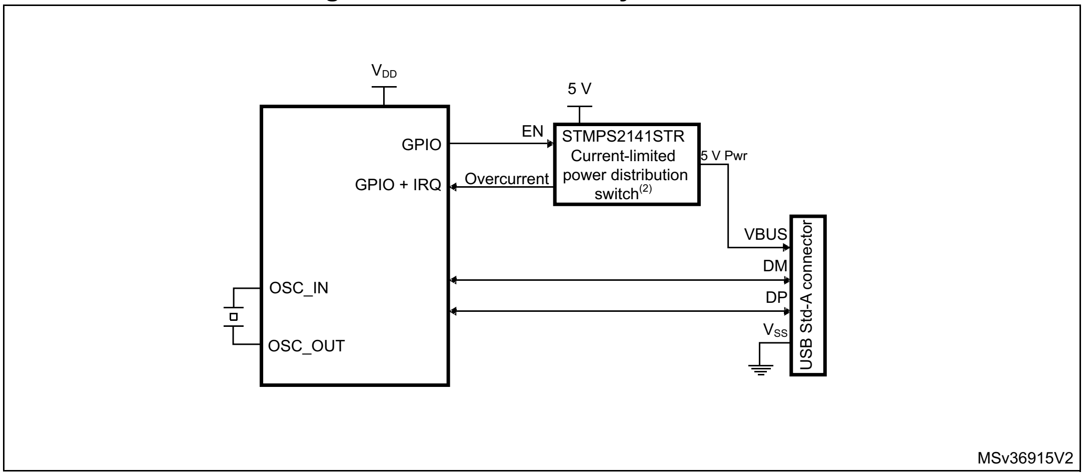
Manual forcing of host mode when not possible via ID pin:

The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG core to work as a USB host-only when required.

- Use cases:
  - Standard-A connector which always implies Host mode.
  - Micro connector, so ID present on connector, but ID pin not available in pinout, at A-plug insertion (detected by another means, for example: GPIO).
  - A Type-C connector, when host functionality is determined at cable attachment or via power delivery messages.
- Note that the integrated pulldown resistors are automatically set on the DP/DM lines

Note: On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive *the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.*

Figure 702. OTG host-only connection



1. V_{DD} range is between 2 V and 3.6 V.

58.7.1 SRP-capable host

SRP support is available through the SRP capable bit in the global USB configuration register (SRPCAP bit in OTG_GUSBCFG). With the SRP feature enabled, the host can save power by switching off the V_{BUS} power while the USB session is suspended.

The SRP host mode program model is described in detail in the *A-device session request protocol*) section.

58.7.2 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output or via an I^2C interface connected to an external PMIC (power management IC). When the application decides to power on V_{BUS}, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

V_{BUS} valid

When HNP or SRP is enabled the VBUS sensing pin must be connected to V_{BUS}. The V_{BUS} input ensures that valid V_{BUS} levels are supplied by the charge pump during USB operations. Any unforeseen V_{BUS} voltage drop below the V_{BUS} valid threshold (4.4 V) leads to an OTG interrupt triggered by the session end detected bit (SEDET bit in OTG_GOTGINT). The application is then required to remove the V_{BUS} power and clear the port power bit.

When HNP and SRP are both disabled, the VBUS sensing pin does not need to be connected to V_{BUS} and it can be used as GPIO.

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Host detection of a peripheral connection

If SRP or HNP are enabled, even if USB peripherals or B-devices can be attached at any time, the OTG does not detect any bus connection until V_{BUS} is no longer sensed at a valid level (5 V). When V_{BUS} is at a valid level and a remote B-device is attached, the OTG core issues a host port interrupt triggered by the device connected bit in the host port control and status register (PCDET bit in OTG_HPRT).

When HNP and SRP are both disabled, USB peripherals or B-device are detected as soon as they are connected. The OTG core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

Before starting to drive a USB reset, the application waits for the OTG interrupt triggered by the debounce done bit (DBCDNE bit in OTG_GOTGINT), which indicates that the bus is stable again after the electrical debounce caused by the attachment of a pull-up resistor on DP (FS) or DM (LS).

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device’s initiative (remote wake-up). In this case the remote wake-up interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wake-up signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

58.7.3 Host channels

The OTG core instantiates 16 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 16 transfer requests at the same time. If more than 16 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of proper control (OTG_HCCHAR*x*), transfer configuration (OTG_HCTSIZ*x*) and status/interrupt (OTG_HCINT*x*) registers with associated mask (OTG_HCINTMSK*x*) registers.

Host channel control

- The following host channel controls are available to the application through the host channel-*x* characteristics register (OTG_HCCHAR*x*):
  - Channel enable/disable
  - Program the HS/FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (OTG_HCTSIZ*x*) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-*x* interrupt register (OTG_HCINT*x*) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

58.7.4 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

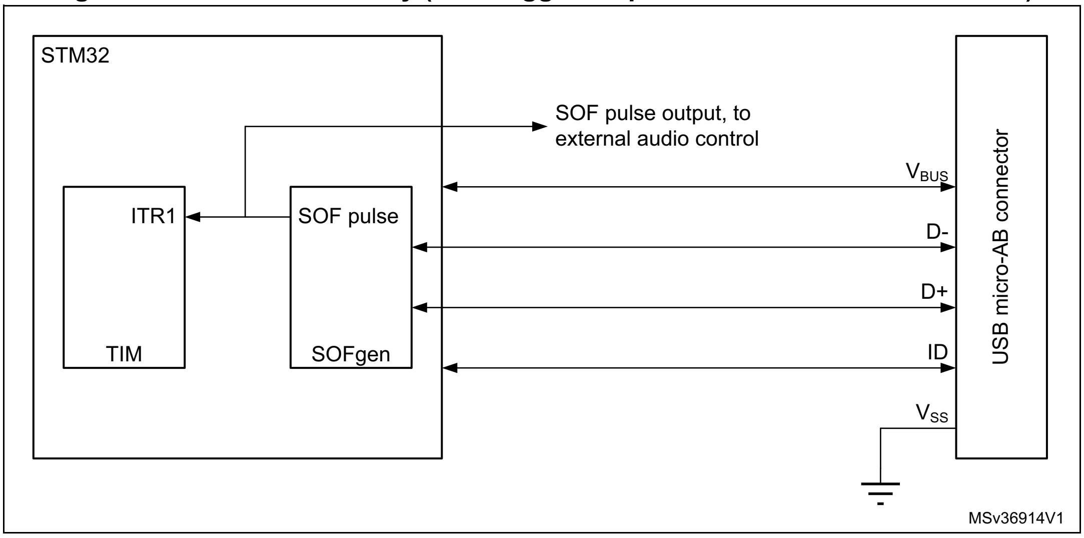
As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the

PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

58.8 OTG SOF trigger

Figure 703. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



The OTG core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

58.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (HS/FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

58.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated.The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

58.9 OTG low-power modes

Table 464 below defines the STM32 low power modes and their compatibility with the OTG.

Table 464. Compatibility of STM32 low power modes with the OTG

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to *Section 9: Power control (PWR)* to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

The power consumption of the OTG PHY is controlled by two or three bits in the general core configuration register, depending on OTG revision supported.

- PHY power down (OTG_GCCFG/PWRDWN)

  It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation
- V_{BUS} detection enable (OTG_GCCFG/VBDEN)

  It switches on/off the V_{BUS} sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)

  When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application

  Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wake-up event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- USB system stop

  When the OTG is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG core automatically reactivates both system and USB clocks by asynchronous detection of remote wake-up (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG core.

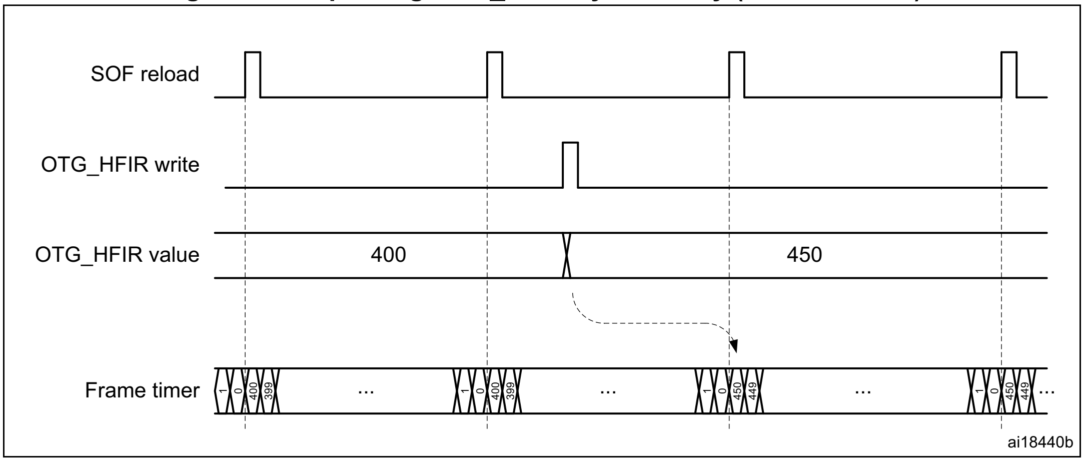
58.10 OTG Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in *Figure 704*.

For a dynamic update, it is required to set RLDCTRL=1.

Figure 704. Updating OTG_HFIR dynamically (RLDCTRL = 1)

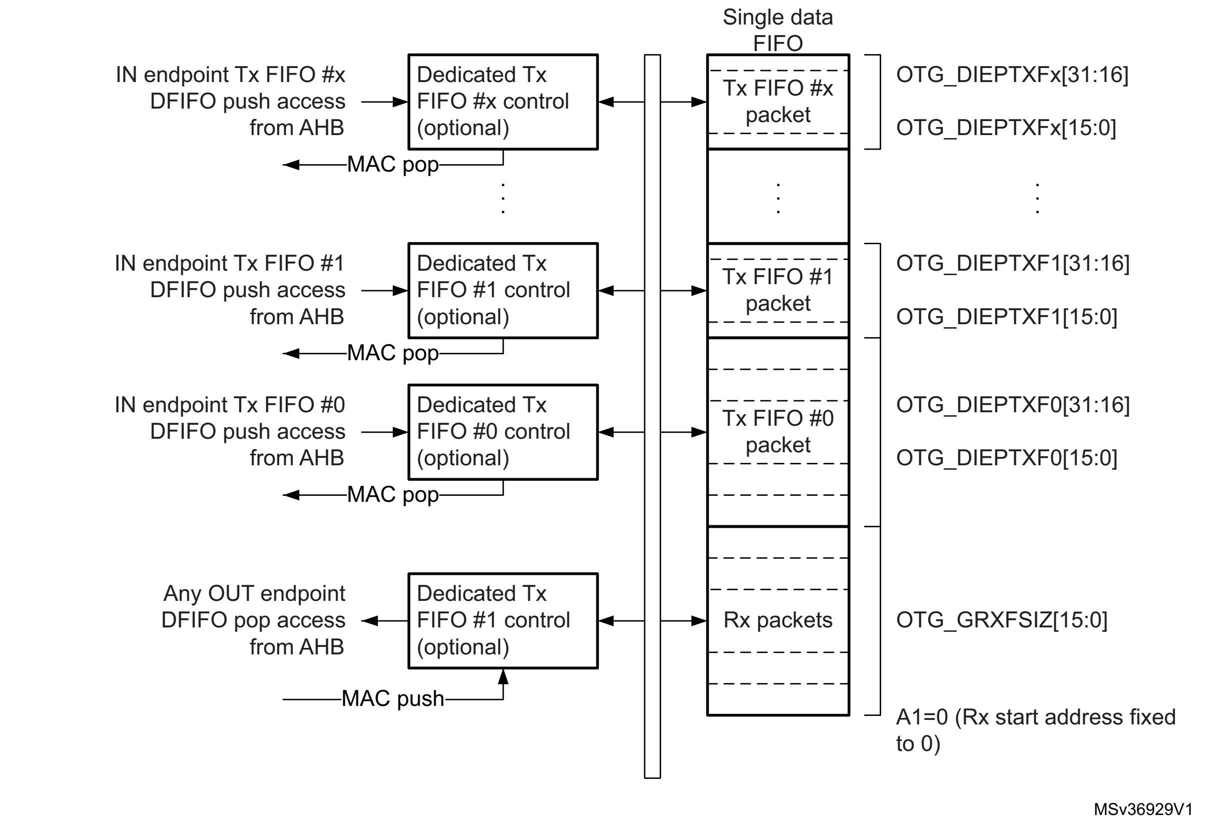


58.11 OTG data FIFOs

The USB system features 4 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device’s role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.

58.11.1 Peripheral FIFO architecture

Figure 705. Device-mode FIFO address mapping and AHB FIFO access mapping



Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

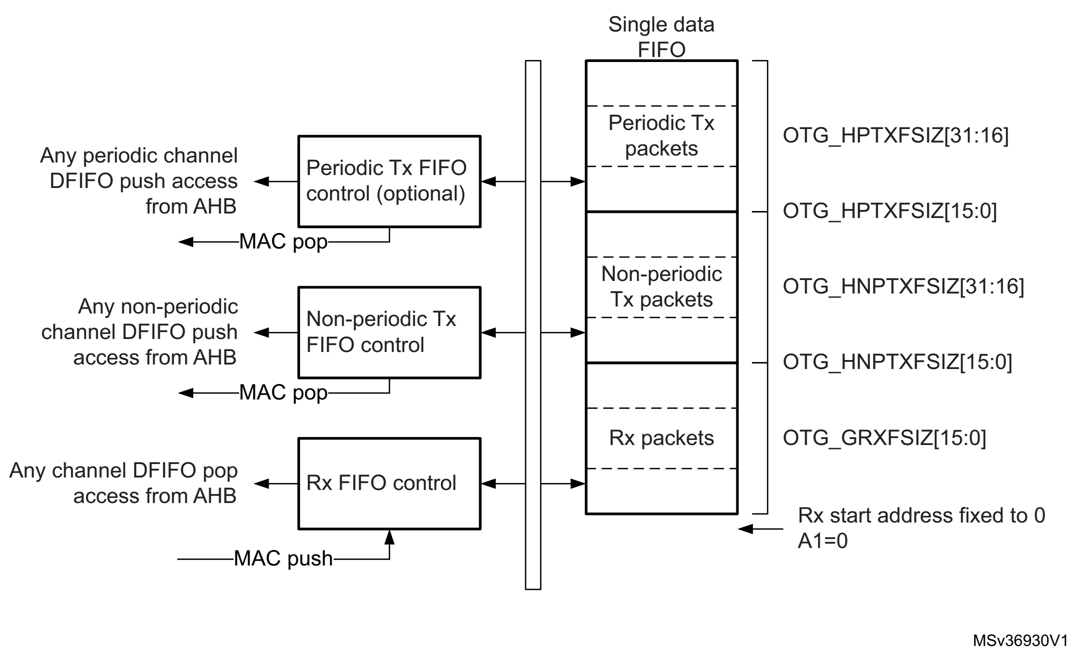
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

58.11.2 Host FIFO architecture

Figure 706. Host-mode FIFO address mapping and AHB FIFO access mapping



Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

58.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application must allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

(5 * number of control endpoints + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = (5 * 1 + 8) + ((1,024 / 4) +1) + (2 * 4) + 1 = 279

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = (largest USB packet used / 4) + 1 for status information + 1 transfer complete

Example: Host RxFIFO = ((1,024 / 4) + 1) + 1 = 258

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = (512 / 4) = 128

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = (1,024 / 4) = 256

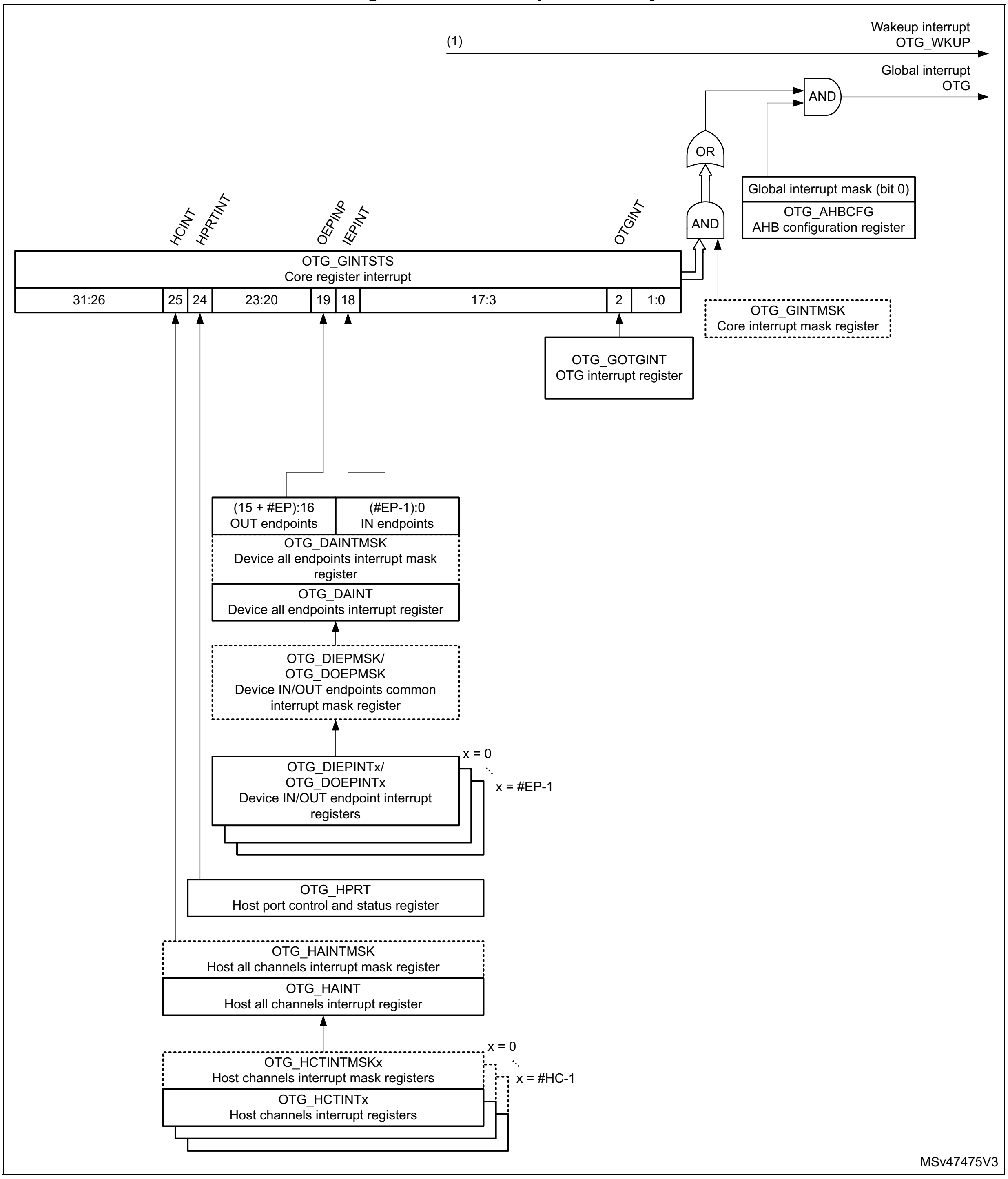
Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.

58.12 OTG interrupts

When the OTG controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 707 shows the interrupt hierarchy.

Figure 707. Interrupt hierarchy



1. OTG_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.
2. See *Table 461: OTG implementation*.

58.13 OTG control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

58.13.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

Table 465. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	*Section 58.14.1: OTG control and status register (OTG_GOTGCTL)*
OTG_GOTGINT	0x004	*Section 58.14.2: OTG interrupt register (OTG_GOTGINT)*
OTG_GAHBCFG	0x008	*Section 58.14.3: OTG AHB configuration register (OTG_GAHBCFG)*
OTG_GUSBCFG	0x00C	*Section 58.14.4: OTG USB configuration register (OTG_GUSBCFG)*
OTG_GRSTCTL	0x010	*Section 58.14.5: OTG reset register (OTG_GRSTCTL)*
OTG_GINTSTS	0x014	*Section 58.14.6: OTG core interrupt register (OTG_GINTSTS)*
OTG_GINTMSK	0x018	*Section 58.14.7: OTG interrupt mask register (OTG_GINTMSK)*

Table 465. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_GRXSTSR	0x01C	*Section 58.14.8: OTG receive status debug read register (OTG_GRXSTSR)*
		Section 58.14.9: OTG receive status debug read [alternate] (OTG_GRXSTSR)
OTG_GRXSTSP	0x020	*Section 58.14.10: OTG status read and pop registers (OTG_GRXSTSP)*
		Section 58.14.11: OTG status read and pop registers [alternate] (OTG_GRXSTSP)
OTG_GRXFSIZ	0x024	*Section 58.14.12: OTG receive FIFO size register (OTG_GRXFSIZ)*
OTG_HNPTXFSIZ/ OTG_DIEPTXF0(1)	0x028	*Section 58.14.13: OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)*
OTG_HNPTXSTS	0x02C	*Section 58.14.14: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)*
OTG_GCCFG	0x038	*Section 58.14.15: OTG general core configuration register (OTG_GCCFG)*
OTG_CID	0x03C	*Section 58.14.16: OTG core ID register (OTG_CID)*
OTG_GLPMCFG	0x54	*Section 58.14.17: OTG core LPM configuration register (OTG_GLPMCFG)*
OTG_HPTXFSIZ	0x100	*Section 58.14.18: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*
OTG_DIEPTXFx	0x104 0x108 ... 0x120	*Section 58.14.19: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)*

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 466. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HCFG	0x400	*Section 58.14.21: OTG host configuration register (OTG_HCFG)*
OTG_HFIR	0x404	*Section 58.14.22: OTG host frame interval register (OTG_HFIR)*
OTG_HFNUM	0x408	*Section 58.14.23: OTG host frame number/frame time remaining register (OTG_HFNUM)*
OTG_HPTXSTS	0x410	*Section 58.14.24: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)*
OTG_HAINT	0x414	*Section 58.14.25: OTG host all channels interrupt register (OTG_HAINT)*
OTG_HAINTMSK	0x418	*Section 58.14.26: OTG host all channels interrupt mask register (OTG_HAINTMSK)*

Table 466. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HFLBADDR	0x41C	*Section 58.14.27: OTG host frame list base address register (OTG_HFLBADDR)*
OTG_HPRT	0x440	*Section 58.14.28: OTG host port control and status register (OTG_HPRT)*
OTG_HCCHARx	0x500 0x520 ... 0x6E0	*Section 58.14.29: OTG host channel x characteristics register (OTG_HCCHARx)*
OTG_HCSPLTx	0x504 0x524 .... 0x6E4	*Section 58.14.30: OTG host channel x split control register (OTG_HCSPLTx)*
OTG_HCINTx	0x508 0x528 .... 0x6E8	*Section 58.14.31: OTG host channel x interrupt register (OTG_HCINTx)*
OTG_HCINTMSKx	0x50C 0x52C 0x6EC	*Section 58.14.32: OTG host channel x interrupt mask register (OTG_HCINTMSKx)*
OTG_HCTSIZx	0x510 0x530 .... 0x6F0	*Section 58.14.33: OTG host channel x transfer size register (OTG_HCTSIZx)*
OTG_HCTSIZSGx	0x510 0x530 .... 0x6F0	*Section 58.14.34: OTG host channel x transfer size register (OTG_HCTSIZSGx)*
OTG_HCDMAx	0x514 0x534 .... 0x6F4	*Section 58.14.35: OTG host channel x DMA address register in buffer DMA [alternate] (OTG_HCDMAx)*
OTG_HCDMASGx	0x514 0x534 .... 0x6F4	*Section 58.14.36: OTG host channel x DMA address register in scatter/gather DMA [alternate] (OTG_HCDMASGx)*
OTG_HCDMABx	0x51C 0x53C 0x6FC	*Section 58.14.37: OTG host channel-n DMA address buffer register (OTG_HCDMABx)*

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 467. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	*Section 58.14.39: OTG device configuration register (OTG_DCFG)*
OTG_DCTL	0x804	*Section 58.14.40: OTG device control register (OTG_DCTL)*
OTG_DSTS	0x808	*Section 58.14.41: OTG device status register (OTG_DSTS)*
OTG_DIEPMSK	0x810	*Section 58.14.42: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)*
OTG_DOEPMSK	0x814	*Section 58.14.43: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)*
OTG_DAINT	0x818	*Section 58.14.44: OTG device all endpoints interrupt register (OTG_DAINT)*
OTG_DAINTMSK	0x81C	*Section 58.14.45: OTG all endpoints interrupt mask register (OTG_DAINTMSK)*
OTG_DVBUSDIS	0x828	*Section 58.14.46: OTG device V_{BUS} discharge time register (OTG_DVBUSDIS)*
OTG_DVBUSPULSE	0x82C	*Section 58.14.47: OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE)*
OTG_DTHRCTL	0x830	*Section 58.14.48: OTG device threshold control register (OTG_DTHRCTL)*
OTG_DIEPEMPMSK	0x834	*Section 58.14.49: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)*
OTG_DIEPCTLx	0x900 0x920 ... 0xA00	*Section 58.14.50: OTG device IN endpoint x control register (OTG_DIEPCTLx)*
OTG_DIEPINTx	0x908 0x928 ... 0x9E8	*Section 58.14.51: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*
OTG_DIEPTSIZ0	0x910	*Section 58.14.52: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)*
OTG_DIEPDMAx	0x914 0x934 ... 0x9F4	*Section 58.14.53: OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)*

Table 467. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DTXFSTSx	0x918 0x938 ..... 0x9F8	*Section 58.14.54: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)*
OTG_DIEPTSIZx	0x930 0x950 ... 0x9F0	*Section 58.14.55: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)*
OTG_DOEPCTL0	0xB00	*Section 58.14.56: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)*
OTG_DOEPINTx	0xB08 0XB28 ... 0xC08	*Section 58.14.57: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)*
OTG_DOEPTSIZ0	0xB10	*Section 58.14.58: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)*
OTG_DOEPDMAx	0xB14 0xB34 ... 0xC14	*Section 58.14.59: OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)*
OTG_DOEPCTLx	0xB20 0xB40 ... 0xC00	*Section 58.14.60: OTG device OUT endpoint x control register (OTG_DOEPCTLx)*
OTG_DOEPTSIZx	0xB30 0xB50 .. 0xBF0	*Section 58.14.61: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)*

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 468. Data FIFO (DFIFO) access register map

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r
...	...	...
Device IN endpoint x[1]/Host OUT Channel x[1]: DFIFO write access Device OUT endpoint x[1]/Host IN Channel x[1]: DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 8in device mode and 15 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

Table 469. Power and clock gating control and status registers

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	*Section 58.14.62: OTG power and clock gating control register (OTG_PCGCCTL)*

58.14 OTG registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

58.14.1 OTG control and status register (OTG_GOTGCTL)

Address offset: 0x000

Reset value: 0x0001 0000

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	DHNP EN	HSHNP EN	HNP RQ	HNG SCS	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	SRQ	SRQ SCS
			rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CURMOD:** Current mode of operation
Indicates the current mode (host or device).
0: Device mode
1: Host mode

Bit 20 **OTGVER:** OTG version
Selects the OTG revision.
0:OTG Version 1.3. OTG1.3 is obsolete for new product development.
1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD:** B-session valid
Indicates the device mode transceiver status.
0: B-session is not valid.
1: B-session is valid.
In OTG mode, the user can use this bit to determine if the device is connected or disconnected.
Note: Only accessible in device mode.

Bit 18 **ASVLD:** A-session valid
Indicates the host mode transceiver status.
0: A-session is not valid
1: A-session is valid
Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time
Indicates the debounce time of a detected connection.
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
1: Short debounce time, used for soft connections (2.5 µs)
Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status
Indicates the connector ID status on a connect event.
0: The OTG controller is in A-device mode
1: The OTG controller is in B-device mode
Note: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN:** Embedded host enable
It is used to select between OTG A device state machine and embedded host state machine.
0: OTG A device state machine is selected
1: Embedded host state machine is selected

Bit 11 **DHNPEN:** Device HNP enabled
The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.
0: HNP is not enabled in the application
1: HNP is enabled in the application
Note: Only accessible in device mode.

Bit 10 **HSHNPEN:** host set HNP enable

The application sets this bit when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) on the connected device.

0: Host Set HNP is not enabled

1: Host Set HNP is enabled

Note: Only accessible in host mode.

Bit 9 **HNPRQ:** HNP request

The application sets this bit to initiate an HNP request to the connected USB host. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.

0: No HNP request

1: HNP request

Note: Only accessible in device mode.

Bit 8 **HNGSCS:** Host negotiation success

The core sets this bit when host negotiation is successful. The core clears this bit when the HNP request (HNPRQ) bit in this register is set.

0: Host negotiation failure

1: Host negotiation success

Note: Only accessible in device mode.

Bit 7 **BVALOVAL:** B-peripheral session valid override value.

This bit is used to set override value for Bvalid signal when BVALOEN bit is set.

0: Bvalid value is '0' when BVALOEN = 1

1: Bvalid value is '1' when BVALOEN = 1

Note: Only accessible in device mode.

Bit 6 **BVALOEN:** B-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.

0:Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core

1:Internally Bvalid received from the PHY is overridden with BVALOVAL bit value

Note: Only accessible in device mode.

Bit 5 **AVALOVAL:** A-peripheral session valid override value.

This bit is used to set override value for Avalid signal when AVALOEN bit is set.

0: Avalid value is '0' when AVALOEN = 1

1: Avalid value is '1' when AVALOEN = 1

Note: Only accessible in host mode.

Bit 4 **AVALOEN:** A-peripheral session valid override enable.

This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.

0:Override is disabled and Avalid signal from the respective PHY selected is used internally by the core

1:Internally Avalid received from the PHY is overridden with AVALOVAL bit value

Note: Only accessible in host mode.

Bit 3 **VBVALOVAL:** V_{BUS} valid override value.
This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.
0: vbusvalid value is '0' when VBVALOEN = 1
1: vbusvalid value is '1' when VBVALOEN = 1
Note: Only accessible in host mode.

Bit 2 **VBVALOEN:** V_{BUS} valid override enable.
This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.
0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core
1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value
Note: Only accessible in host mode.

Bit 1 **SRQ:** Session request
The application sets this bit to initiate a session request on the USB. The application can clear this bit by writing a 0 when the host negotiation success status change bit in the OTG_GOTGINT register (HNSSCHG bit in OTG_GOTGINT) is set. The core clears this bit when the HNSSCHG bit is cleared.
If the user uses the USB 1.1 full-speed serial transceiver interface to initiate the session request, the application must wait until V_{BUS} discharges to 0.2 V, after the B-session valid bit in this register (BSVLD bit in OTG_GOTGCTL) is cleared.
0: No session request
1: Session request
Note: Only accessible in device mode.

Bit 0 **SRQSCS:** Session request success
The core sets this bit when a session request initiation is successful.
0: Session request failure
1: Session request success
Note: Only accessible in device mode.

58.14.2 OTG interrupt register (OTG_GOTGINT)

Address offset: 0x04

Reset value: 0x0000 0000

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBCDNE	ADTOCHG	HNGDET	Res.
												rc_w1	rc_w1	rc_w1	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HNSSCHG	SRSSCHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
						rc_w1	rc_w1						rc_w1		

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **DBCDNE:** Debounce done

The core sets this bit when the debounce is completed after the device connect. The application can start driving USB reset after seeing this interrupt. This bit is only valid when the HNP Capable or SRP Capable bit is set in the OTG_GUSBCFG register (HNPCAP bit or SRPCAP bit in OTG_GUSBCFG, respectively).

Note: Only accessible in host mode.

Bit 18 **ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both device and host modes.

Bit 17 **HNGDET:** Host negotiation detected

The core sets this bit when it detects a host negotiation request on the USB.

Note: Accessible in both device and host modes.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **HNSSCHG:** Host negotiation success status change

The core sets this bit on the success or failure of a USB host negotiation request. The application must read the host negotiation success bit of the OTG_GOTGCTL register (HNGSCS bit in OTG_GOTGCTL) to check for success or failure.

Note: Accessible in both device and host modes.

Bits 7:3 Reserved, must be kept at reset value.

Bit 8 **SRSSCHG:** Session request success status change

The core sets this bit on the success or failure of a session request. The application must read the session request success bit in the OTG_GOTGCTL register (SRQSCS bit in OTG_GOTGCTL) to check for success or failure.

Note: Accessible in both device and host modes.

Bit 2 **SEDET:** Session end detected

The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-Peripheral session when V_{BUS} < 0.8 V.

Note: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.

58.14.3 OTG AHB configuration register (OTG_GAHBCFG)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFE LVL	TXFE LVL	Res.	DMAEN	HBSTLEN[3:0]				GINT MSK
							rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.

0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** Tx FIFO empty level

In device mode, this bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:

0:The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1:The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

In host mode, this bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:

0:The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1:The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bit 6 Reserved, must be kept at reset value.

Bit 5 **DMAEN:** DMA enabled

0: The core operates in slave mode
1: The core operates in DMA mode

Bits 4:1 **HBSTLEN[3:0]:** Burst length/type

0000 Single: Bus transactions use single 32 bit accesses (not recommended)
0001 INCR: Bus transactions use unspecified length accesses (not recommended, uses the INCR AHB bus command)
0011 INCR4: Bus transactions target 4x 32 bit accesses
0101 INCR8: Bus transactions target 8x 32 bit accesses

0111 INCR16: Bus transactions based on 16x 32 bit accesses
Others: Reserved

Bit 0 **GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.
0: Mask the interrupt assertion to the application.
1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

58.14.4 OTG USB configuration register (OTG_GUSBCFG)

Address offset: 0x00C

Reset value: 0x0000 1400

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD MOD	FH MOD	Res.	Res.	Res.	Res.	Res.	Res.	TSDPS	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw							rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYL PC	Res.	TRDT[3:0]				HNP CAP	SRP CAP	Res.	PHY SEL	Res.	Res.	Res.	TOCAL[2:0]		
rw		rw	rw	rw	rw	rw	rw		rw				rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FDMOD:** Force device mode

Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force device mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bit 29 **FHMOD:** Force host mode

Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force host mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 **TSDPS**: TermSel DLine pulsing selection

This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).
0: Data line pulsing using utmi_txvalid (default)
1: Data line pulsing using utmi_termsel

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 **PHYLPC:** PHY Low-power clock select

This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.
0: 480 MHz internal PLL clock
1: 48 MHz external clock
In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]:** USB turnaround time

These bits are used to set the turnaround time in PHY clocks. They must be configured according to *Table 470: TRDT values*, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

Note: Only accessible in device mode.

Bit 9 **HNPCAP:** HNP-capable

The application uses this bit to control the OTG controller’s HNP capabilities.
0: HNP capability is not enabled.
1: HNP capability is enabled.

Note: Accessible in both device and host modes.

Bit 8 **SRPCAP:** SRP-capable

The application uses this bit to control the OTG controller's SRP capabilities. If the core operates as a non-SRP-capable
B-device, it cannot request the connected A-device (host) to activate V_{BUS} and start a session.

0: SRP capability is not enabled.

1: SRP capability is enabled.

Note: Accessible in both device and host modes.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PHYSEL:** Full speed serial transceiver mode select

0: USB 2.0 internal UTMI high-speed PHY.

1: USB 1.1 full-speed serial mode. Refer also to PWRDWN setting which specifies the destination PHY for this full speed serial traffic.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

Table 470. TRDT values

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
30	-	0x9

58.14.5 OTG reset register (OTG_GRSTCTL)

Address offset: 0x10

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHBIDL	DMAREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM[4:0]					TXFFLSH	RXFFLSH	Res.	FCRST	PSRST	CSRST
					rw	rw	rw	rw	rw	rs	rs		rs	rs	rs

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both device and host modes.

Bit 30 **DMAREQ:** DMA request signal enabled

This bit indicates that the DMA request is in progress. Used for debug.

Bits 29:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]:** Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

00000:

- Non-periodic Tx FIFO flush in host mode
- Tx FIFO 0 flush in device mode

00001:

- Periodic Tx FIFO flush in host mode
- Tx FIFO 1 flush in device mode

00010: Tx FIFO 2 flush in device mode

...

01111: Tx FIFO 15 flush in device mode

10000: Flush all the transmit FIFOs in device or host mode.

Note: Accessible in both device and host modes.

Bit 5 **TXFFLSH:** Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

Note: Accessible in both device and host modes.

Bit 4 **RXFFLSH:** Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FCRST:** Host frame counter reset

The application writes this bit to reset the (micro-)frame number counter inside the core. When the (micro-)frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.

When application writes '1' to the bit, it might not be able to read back the value as it gets cleared by the core in a few clock cycles.

Note: Only accessible in host mode.

Bit 1 **PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Can be used to recover some specific PHY errors.

Note: Accessible in both device and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

– GATEHCLK bit in OTG_PCGCCTL

– STPPCLK bit in OTG_PCGCCTL

– FSLSPCS bits in OTG_HCFG

– DSPD bit in OTG_DCFG

– SDIS bit in OTG_DCTL

– OTG_GCCFG register

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.

Typically, the software reset is used during software development and also when the user dynamically changes the PHY selection bits in the above listed USB configuration registers. When the user changes the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both device and host modes.

58.14.6 OTG core interrupt register (OTG_GINTSTS)

Address offset: 0x014

Reset value: 0x0400 0020

This register interrupts the application for system-level events in the current mode (device mode or host mode).

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	DATAF SUSP	IPXFR/ IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1	rc_w1	rc_w1	rc_w1	r	r		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wake-up detected interrupt

Wake-up interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

INCOMPISOOUT: Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode
– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.
0: Device mode
1: Host mode

Note: Accessible in both host and device modes.

58.14.7 OTG interrupt mask register (OTG_GINTMSK)

Address offset: 0x018

Reset value: 0x0000 0000

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wake-up detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both host and device modes.

Bit 26 **PTXFEM:** Periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 23 **RSTDETM:** Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 22 **FSUSPM:** Data fetch suspended mask
0: Masked interrupt
1: Unmasked interrupt
Only accessible in peripheral mode.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.
IISOOXFRM: Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 6 **GINAKEFFM:** Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in device mode.

Bit 5 **NPTXFEM:** Non-periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Only accessible in host mode.

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt
Note: Accessible in both device and host modes.

Bit 0 Reserved, must be kept at reset value.

58.14.8 OTG receive status debug read register (OTG_GRXSTSR)

Address offset for read: 0x01C

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

58.14.9 OTG receive status debug read [alternate] (OTG_GRXSTSR)

Address offset for read: 0x01C

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

58.14.10 OTG status read and pop registers (OTG_GRXSTSP)

Address offset for pop: 0x020

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start

Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number
Indicates the endpoint number to which the current received packet belongs.

58.14.11 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Address offset for pop: 0x020

Reset value: 0x0000 0000

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

58.14.12 OTG receive FIFO size register (OTG_GRXFSIZ)

Address offset: 0x024

Reset value: 0x0000 0400

The application can program the RAM size that must be allocated to the Rx FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFD[15:0]:** Rx FIFO depth
This value is in terms of 32-bit words.
Maximum value is 1024
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

58.14.13 OTG host non-periodic transmit FIFO size register (OTG_HNPTXFSIZ)/Endpoint 0 Transmit FIFO size (OTG_DIEPTXF0)

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD/TX0FD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NPTXFSA/TX0FSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Host mode

Bits 31:16 **NPTXFD[15:0]:** Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]:** Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

Device mode

Bits 31:16 **TX0FD:** Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA:** Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

58.14.14 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Address offset: 0x02C

Reset value: 0x0008 0400

Note: In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]:** Top of the non-periodic transmit request queue
Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
00: IN/OUT token
01: Zero-length transmit packet (device IN/host OUT)
11: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]:** Non-periodic transmit request queue space available
Indicates the amount of free space available in the non-periodic transmit request queue.
This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available ($0 \leq n \leq 8$)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]:** Non-periodic Tx FIFO space available
Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where $0 \leq n \leq 512$)
Others: Reserved

58.14.15 OTG general core configuration register (OTG_GCCFG)

Address offset: 0x038

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDEN	VBDEN	SDEN	PDEN	Res.	BCDEN	PWR DWN
									rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2 DET	SDET	PDET	Res.
												r	r	r	

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **IDEN**: USB ID detection enable

Enables detection of the ID pin state.

0: Disable ID pin state detection

1: Enable ID pin state detection

Bit 21 **VBDEN:** USB V_{BUS} detection enable

Enables V_{BUS} sensing comparators to detect V_{BUS} valid levels on the V_{BUS} PAD for USB host and device operation. If HNP and/or SRP support is enabled, V_{BUS} comparators are automatically enabled independently of VBDEN value.

0 = V_{BUS} detection disabled

1 = V_{BUS} detection enabled

Bit 20 **SDEN:** Secondary detection (SD) mode enable

This bit is set by the software to put the BCD into SD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly

Bit 19 **PDEN:** Primary detection (PD) mode enable

This bit is set by the software to put the BCD into PD mode. Only one detection mode (PD, SD or OFF) must be selected to work correctly.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **BCDEN:** Battery charging detector (BCD) enable

This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to '0' in order to allow the normal USB operation.

Bit 16 **PWRDWN:** Power down control of FS PHY and selection of HS PHY when PHYSEL=1

Used to activate the FS PHY in transmission/reception. When reset, the PHY is kept in power-down. When set, the BCD function must be off (BCDEN=0).

0 = USB FS PHY disabled; HS PHY enabled

1 = USB FS PHY enabled; HS PHY disabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PS2DET:** DM pull-up detection status

This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, CDP or DCP)

1: PS2 port or proprietary charger detected

Bit 2 **SDET:** Secondary detection (SD) status

This bit gives the result of SD.

0: CDP detected

1: DCP detected

Bit 1 **PDET:** Primary detection (PD) status

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to CDP or DCP).

Bit 0 Reserved, must be kept at reset value.

58.14.16 OTG core ID register (OTG_CID)

Address offset: 0x03C

Reset value: 0x0000 4000

This is a register containing the Product ID as reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCT_ID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRODUCT_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT_ID[31:0]:** Product ID field
Application-programmable ID field.

58.14.17 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]				L1RSM OK
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	rw	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL:** Enable best effort service latency
This bit enables the BESL feature as defined in the LPM errata:
0:The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
1:The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007

Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit must be set to '1' by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]:** LPM retry count status
Number of LPM host retries still remaining to be transmitted for the current LPM sequence.

Note: Accessible only in host mode.

Bit 24 **SNDLPM:** Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

Note: This bit must be set only when the host is connected to a local port.

Note: Accessible only in host mode.

Bits 23:21 **LPMRCNT[2:0]:** LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

Note: Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]:** LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

Note: Accessible only in host mode.

Bit 16 **L1RSMOK:** Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 µs ($T_{L1Residency}$).

This bit is reset when SLPSTS is 0.

1: The application or host can start resume from Sleep state

0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS:** Port sleep status

Device mode:

This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1TokenRetry}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.

The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.

The core comes out of sleep:

– When there is any activity on the USB line state

– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

Host mode:

The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.

The core clears this bit after:

– The core detects a remote L1 wake-up signal,

– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or

– The application sets the L1Resume/ remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).

0: Core not in L1

1: Core in L1

Bits 14:13 **LPMRSP[1:0]:** LPM response

Device mode:

The response of the core to LPM transaction received is reflected in these two bits.

Host mode:

Handshake response received from local device for LPM transaction

11: ACK

10: NYET

01: STALL

00: ERROR (No handshake response)

Bit 12 **L1DSEN:** L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bits 11:8 **BESLTHRS[3:0]:** BESL threshold

Device mode:

The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

Host mode:

The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.

Thres[3:0] host mode resume signaling time (µs):

0000: 75

0001: 100

0010: 150

0011: 250

0100: 350

0101: 450

0110: 950

All other values: reserved

Bit 7 **L1SSEN:** L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE:** bRemoteWake value

Host mode:

The value of remote wake up to be sent in the wIndex field of LPM transaction.

Device mode (read-only):

This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]:** Best effort service latency

Host mode:

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{L1HubDrvResume1}$ for host initiated resume.

Device mode (read-only):

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0]T_{BESL} (µs)

0000: 125
0001: 150
0010: 200
0011: 300
0100: 400
0101: 500
0110: 1000
0111: 2000
1000: 3000
1001: 4000
1010: 5000
1011: 6000
1100: 7000
1101: 8000
1110: 9000
1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

- No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
- Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
- No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

- The received bLinkState is not L1 (STALL response), or
- An error is detected in either of the LPM token packets because of corruption (ERROR response).

Note: Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

The application uses this bit to control the OTG core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled

1: LPM capability is enabled

58.14.18 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0400 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXFSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PTXFSIZ[15:0]:** Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]:** Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

58.14.19 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 8)

Reset value: Block 1: 0x0200 0400

Reset value: Block 2: 0x0200 0600

Reset value: Block 3: 0x0200 0800

Reset value: Block 4: 0x0200 0A00

Reset value: Block 5: 0x0200 0C00

Reset value: Block 6: 0x0200 0E00

Reset value: Block 7: 0x0200 1000

Reset value: Block 8: 0x0200 1200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INEPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INEPTXFD[15:0]:** IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

58.14.20 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

58.14.21 OTG host configuration register (OTG_HCFG)

Address offset: 0x400

Reset value: 0x0000 0000

This register configures the core after power-on. Do not make changes to this register after initializing the host.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PERSSCHEDENA	FRLSTEN[1:0]		DESCDMA	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLSPCS[1:0]	
													r	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PERSSCHEDENA**: Enable periodic scheduling
Applicable in host scatter/gather DMA mode only. Enables periodic scheduling within the core. Initially, the bit is res and the core does not process any periodic channels. As soon as this bit is set, the core gets ready to start scheduling periodic channels and sets OTG_HCFG.PERSCHEDSTAT. The setting of PERSCHEDSTAT indicates the core has enabled periodic scheduling. Once PERSSCHEDENA is set, the application is not supposed to reset the bit unless PERSCHEDSTAT is set. As soon as this bit is reset, the core gets ready to stop scheduling periodic channels and resets HCFG. PerSchedStat. In non-Scatter/Gather DMA mode, this bit is reserved.

Bits 25:24 **FRLSTEN[1:0]**: Frame list entries
The value in the register specifies the number of entries in the Frame list. This field is valid only in Scatter/Gather DMA mode.
2'b00: Reserved
2'b01: 8 Entries
2'b10: 16 Entries
2'b11: 32 Entries In non-Scatter/Gather

Bit 23 **DESCDMA**: Enable scatter/gather DMA in host mode

The application can set this bit during initialization to enable the Scatter/Gather DMA operation. This bit must be modified only once after a reset. The following combinations are available for programming:
OTG_GAHBCFG.DMAEN=0,OTG_HCFG.DESCDMA=0 => Slave mode
OTG_GAHBCFG.DMAEN=0,OTG_HCFG.DESCDMA=1 => Invalid
OTG_GAHBCFG.DMAEN=1,OTG_HCFG.DESCDMA=0 => Buffered DMA mode
OTG_GAHBCFG.DMAEN=1,OTG_HCFG.DESCDMA=1 => Scatter/Gather DMA mode

Bits 22:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support

The application uses this bit to control the core's enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

Bits 1:0 **FSLSPCS[1:0]:** FS/LS PHY clock select

When the core is in FS host mode
01: PHY clock is running at 48 MHz
Others: Reserved
When the core is in LS host mode
00: Reserved
01: Select 48 MHz PHY clock frequency
10: Select 6 MHz PHY clock frequency
11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

58.14.22 OTG host frame interval register (OTG_HFIR)

Address offset: 0x404

Reset value: 0x0000 EA60

This register stores the frame interval information for the current speed to which the OTG controller has enumerated.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLD CTRL
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRIVL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RLDCTRL:** Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR cannot be reloaded dynamically

1: The HFIR can be dynamically reloaded during run time.

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

Caution: RLDCTRL = 0 is not recommended.

Bits 15:0 **FRIVL[15:0]:** Frame interval

The value that the application programs to this field, specifies the interval between two consecutive micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

– Frame interval = 125 µs × (FRIVL - 1) in high speed operation

– Frame interval = 1 ms × (FRIVL - 1) in low/full speed operation

58.14.23 OTG host frame number/frame time remaining register (OTG_HFNUM)

Address offset: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTREM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FRNUM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FTREM[15:0]:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

58.14.24 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

Address offset: 0x410

Reset value: 0x0008 0100

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame

0: send in even frame

1: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type

00: IN/OUT

01: Zero-length packet

11: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

00: Periodic transmit request queue is full

01: 1 location available

10: 2 locations available

bxn: n locations available ($0 \le n \le 8$)

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0000: Periodic Tx FIFO is full

0001: 1 word available

0010: 2 words available

bxn: n words available (where $0 \le n \le$ PTXFD)

Others: Reserved

58.14.25 OTG host all channels interrupt register (OTG_HAINT)

Address offset: 0x414

Reset value: 0x0000 0000

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in *Figure 707*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HAINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINT[15:0]:** Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

58.14.26 OTG host all channels interrupt mask register (OTG_HAINTMSK)

Address offset: 0x418

Reset value: 0x0000 0000

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HAINTM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM[15:0]:** Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

58.14.27 OTG host frame list base address register (OTG_HFLBADDR)

Address offset: 0x41C

Reset value: 0x0000 0000

This register holds the starting address of the frame list information (scatter/gather mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HFLBADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HFLBADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HFLBADDR[31:0]**: The starting address of the frame list (scatter/gather mode).
This register is used only for isochronous and interrupt channels.

58.14.28 OTG host port control and status register (OTG_HPRT)

Address offset: 0x440

Reset value: 0x0000 0000

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 707*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL [3]
													r	r	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]:** Port speed

Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved
00: High speed

Bits 16:13 **PTCTL[3:0]:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition.
0: Power off
1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status

Indicates the current logic level USB data lines
Bit 10: Logic level of OTG_DP
Bit 11: Logic level of OTG_DM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.
0: Port not in reset
1: Port in reset
The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.
High speed: 50 ms
Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.
The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wake-up signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).
0: Port not in suspend mode
1: Port in suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wake-up sequence, as indicated by the port resume/remote wake-up detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.

3. If the core detects a USB remote wake-up sequence, as indicated by the port L1Resume/Remote L1Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume.This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

58.14.29 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: 0x500 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPSIZ[10:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

When Scatter/Gather mode is enabled:

1'b0: Indicates that the descriptor structure is not yet ready

1'b1: Indicates that the descriptor structure and data buffer with data is setup and this channel can access the descriptor

When Scatter/Gather mode is disabled: This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame

1: Odd frame

Bits 28:22 **DAD[6:0]:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]:** Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

00: Reserved. This field yields undefined results

01: 1 transaction

10: 2 transactions per frame to be issued for this endpoint

11: 3 transactions per frame to be issued for this endpoint

Note: This field must be set to at least 01.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Indicates the transfer type selected.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.
0: OUT
1: IN

Bits 14:11 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

58.14.30 OTG host channel x split control register (OTG_HCSPLTx)

Address offset: 0x504 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLIT EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP LSPLT
rw															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
XACTPOS[1:0]		HUBADDR[6:0]							PRTADDR[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPLITEN:** Split enable

The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT:** Do complete split

The application sets this bit to request the OTG host to perform a complete split transaction.

Bits 15:14 **XACTPOS[1:0]:** Transaction position

This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.
11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)
10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)
00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)
01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7 **HUBADDR[6:0]:** Hub address

This field holds the device address of the transaction translator's hub.

Bits 6:0 **PRTADDR[6:0]:** Port address

This field is the port number of the recipient transaction translator.

58.14.31 OTG host channel x interrupt register (OTG_HCINTx)

Address offset: 0x508 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 707*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DESCLSTROLL	XCSXACTERR	BNA	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DESCLSTROLL:** Descriptor rollover interrupt.

This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when the corresponding channel descriptor list rolls over. For non Scatter/Gather DMA mode, this bit is reserved.

Bit 12 **XCSXACTERR:** Excessive transaction error.

This bit is valid only when Scatter/Gather DMA mode is enabled. The core sets this bit when 3 consecutive transaction errors occurred on the USB bus. XCS_XACT_ERR is not generated for isochronous channels. For non Scatter/Gather DMA mode, this bit is reserved.

Bit 11 **BNA:** Buffer not available interrupt.

This bit is valid only when Scatter/Gather DMA mode is enabled. The core generates this interrupt when the descriptor accessed is not ready for the core to process. BNA interrupt is not generated for isochronous channels. For non Scatter/Gather DMA mode, this bit is reserved.

Bit 10 **DTERR:** Data toggle error. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 9 **FRMOR:** Frame overrun. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 8 **BBERR:** Babble error. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 7 **TXERR:** Transaction error. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Indicates one of the following errors occurred on the USB.
CRC check failure
Timeout
Bit stuff error
False EOP

Bit 6 **NYET:** Not yet ready response received interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 5 **ACK:** ACK response received/transmitted interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 4 **NAK:** NAK response received interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 3 **STALL:** STALL response received interrupt. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.

Bit 2 **AHBERR:** AHB error

This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1 **CHH:** Channel halted

In non scatter/gather DMA mode, this indicates that the transfer completed:
- because of any USB transaction error
- in response to disable request by the application
- normally

In scatter/gather DMA mode, this indicates that the transfer completed due to any of the following:
- EOL being set in descriptor
- AHB error
- Excessive transaction errors
- In response to disable request by the application
- Babble
- Stall

Bit 0 **XFRC:** Transfer completed.

Transfer completed normally without any errors.

58.14.32 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

Address offset: 0x50C + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

This register reflects the mask for each channel status described in the previous section.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	DESCLSTROLLMSK	Res.	BNAMSK	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **DESCLSTROLLMSK:** Descriptor rollover interrupt mask register.
This bit is valid only when Scatter/Gather DMA mode is enabled.
In non Scatter/Gather DMA mode, this bit is reserved.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **BNAMSK:** Buffer not available interrupt mask register.
This bit is valid only when Scatter/Gather DMA mode is enabled.
In non Scatter/Gather DMA mode, this bit is reserved

Bit 10 **DTERRM:** Data toggle error mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 6 **NYET:** response received interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask. In Scatter/Gather DMA mode, the interrupt due to this bit is masked.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error. In scatter/gather DMA mode for host, interrupts will not be generated due to the corresponding bits set in OTG_HCINTx.
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **CHHM:** Channel halted mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask
0: Masked interrupt
1: Unmasked interrupt

58.14.33 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOPNG	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPNG:** Do Ping
This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.
Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **DPID[1:0]:** Data PID
The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.
00: DATA0
01: DATA2
10: DATA1
11: SETUP (control) / MDATA (non-control)

Bits 28:19 **PKTCNT[9:0]:** Packet count
This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).
The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size
For an OUT, this field is the number of data bytes the host sends during the transfer.
For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

58.14.34 OTG host channel x transfer size register (OTG_HCTSIZSGx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DO PNG	PID[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTD[7:0]								SCHED_INFO[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPNG:** Do Ping

This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **PID[1:0]:** Pid

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

00: DATA0

01: DATA2

10: DATA1

11: MDATA (non-control) / SETUP (control)

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:8 **NTD[7:0]:** Number of transfer descriptors

Non isochronous: this value is in terms of number of descriptors. The maximum number of descriptor that can be present in the list is 64. The values can be from 0 to 63.
0: 1 descriptor
1: 2 descriptors
...
63: 64 descriptors
Others: Reserved
This field indicates the total number of descriptors present in that list. The core will wrap around after servicing NTD number of descriptors for that list.

Isochronous: this field indicates the number of descriptors present in that list.
The possible values for FS are:
1: 2 descriptors
3: 4 descriptors
7: 8 descriptors
15: 16 descriptors
31: 32 descriptors
63: 64: descriptors
Others: Reserved
The possible values for HS are:
7: 8 descriptors
15: 16 descriptors
31: 32 descriptors
63: 64 descriptors
127: 128 descriptors
255: 256 descriptors
Others: Reserved

Bits 7:0 **SCHED_INFO[7:0]**: Schedule information

Every bit in this 8 bit register indicates scheduling for that microframe. Bit 0 indicates scheduling for 1st microframe and bit 7 indicates scheduling for 8th microframe in that frame. A value of 0b11111111 indicates that the corresponding interrupt channel is scheduled to issue a token every microframe in that frame. A value of 0b10101010 indicates that the corresponding interrupt channel is scheduled to issue a token every alternate microframe starting with second microframe. Note that this field is applicable only for periodic (isochronous and interrupt) channels.

58.14.35 OTG host channel x DMA address register in buffer DMA [alternate] (OTG_HCDMAx)

Address offset: 0x514 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA address

This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

58.14.36 OTG host channel x DMA address register in scatter/gather DMA [alternate] (OTG_HCDMASGx)

Address offset: 0x514 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMASG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMASG[15:3]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:3 **DMASG[31:3]:** DMA scatter/gather information

The DMASG information is composed of two parts DMASG_ADDR = DMASG[31:N] and DMASG_CTD = DAMSG[N-1:3].

Non-isochronous case (N = 9):

The DAMSG_ADDR field holds the start address of the 512 bytes page. The first descriptor in the list should be located in this address. The first descriptor may be or may not be ready. The core starts processing the list from the CTD value.

The DMASG_CTD field holds a value in terms of number of descriptors. The values can be from 0 (1 descriptor) to 63 (64 descriptors). This field indicates the current descriptor processed in the list. This field is updated both by application and the core. For example, if the application enables the channel after programming CTD = 5, then the core will start processing the 6th descriptor. The address is obtained by adding a value of 8 * 5 = 40 bytes (decimal) to DMAAddr.

Isochronous case (N=4 to 11):

The DAMSG_ADDR field holds the address of the 2*(NTD+1) bytes of locations in which the isochronous descriptors are present where N is based on NTD as per the following table:

HS ISOC; NTD = 7:	N = 6
HS ISOC; NTD = 15:	N = 7
HS ISOC; NTD = 31:	N = 8
HS ISOC; NTD = 63:	N = 9
HS ISOC; NTD = 127:	N = 10
HS ISOC; NTD = 255:	N = 11
FS ISOC; NTD = 1:	N = 4
FS ISOC; NTD = 3:	N = 5
FS ISOC; NTD = 7:	N = 6
FS ISOC; NTD = 15:	N = 7
FS ISOC; NTD = 31:	N = 8
FS ISOC; NTD = 63:	N = 9

The DMASG_CTD field is based on the current frame/(micro)frame value. Need to be set to zero by application.

Bits 2:0 Reserved, must be kept at reset value.

58.14.37 OTG host channel-n DMA address buffer register (OTG_HCDMABx)

Address offset: 0x51C + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000 (0x0000 0000)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HCDMAB[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HCDMAB[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **HCDMAB[31:0]:** DMA address

This register holds the current buffer address (scatter/gather mode).

58.14.38 Device-mode registers

These registers must be programmed every time the core changes to device mode

58.14.39 OTG device configuration register (OTG_DCFG)

Address offset: 0x800

Reset value: 0x0220 0000

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL[1:0]		DESCDMA	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw	rw							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ERRATIM	Res.	Res.	PFIVL[1:0]		DAD[6:0]							Res.	NZLSOHSK	DSPD[1:0]	
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **PERSCHIVL[1:0]:** Periodic schedule interval

This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro) frame.

- When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data
- When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field
- After the specified time within a (micro) frame, the DMA switches to fetching nonperiodic endpoints

00: 25% of (micro)frame
01: 50% of (micro)frame
10: 75% of (micro)frame
11: Reserved

Note: Periodic Scheduling Interval (PERSCHIVL) must be programmed for Scatter/Gather DMA mode.

Bit 23 **DESCDMA:** Enable scatter/gather DMA in device mode

The application can set this bit during initialization to enable the Scatter/Gather DMA operation. This bit must be modified only once after a reset. The following combinations are available for programming:

OTG_GAHBCFG.DMAEN=0,OTG_DCFG.DESCDMA=0 => Slave mode
OTG_GAHBCFG.DMAEN=0,OTG_DCFG.DESCDMA=1 => Invalid
OTG_GAHBCFG.DMAEN=1,OTG_DCFG.DESCDMA=0 => Buffer DMA mode
OTG_GAHBCFG.DMAEN=1,OTG_DCFG.DESCDMA=1 => Scatter/Gather DMA mode

Bits 22:16 Reserved, must be kept at reset value.

Bit 15 **ERRATIM:** Erratic error interrupt mask
1: Mask early suspend interrupt on erratic error
0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval
Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.
00: 80% of the frame interval
01: 85% of the frame interval
10: 90% of the frame interval
11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address
The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake
The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.
1:Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
0:Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]:** Device speed
Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.
00: High speed
01: Full speed using HS
10: Reserved
11: Full speed using internal FS PHY

58.14.40 OTG device control register (OTG_DCTL)

Address offset: 0x804

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DS BESL RJCT	ENCO NTONB NA	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PO PRG DNE	CGO NAK	SGO NAK	CGI NAK	SGI NAK	TCTL[2:0]			GON STS	GIN STS	SDIS	RWU SIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT:** Deep sleep BESL reject

Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bit 17 ENCONTONBNA: Enable continue on BNA

This bit enables the core to continue on BNA for Bulk OUT and INTR OUT endpoints. With this feature enabled, when a Bulk OUT or INTR OUT endpoint receives a BNA interrupt the core starts processing the descriptor that caused the BNA interrupt after the endpoint re-enables the endpoint.

0: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the DOEPDMA descriptor.

1: After receiving BNA interrupt, the core disables the endpoint. When the endpoint is re-enabled by the application, the core starts processing from the descriptor that received the BNA interrupt. It is a one-time programmable after reset bit like any other OTG_DCTL register bit.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done

The application uses this bit to indicate that register programming is completed after a wake-up from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK

Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]:** Test control

000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status

0:A handshake is sent based on the FIFO status and the NAK and STALL bit settings.
1:No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status

0:A handshake is sent out based on the data availability in the transmit FIFO.

1:A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.

0:Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.

1:The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wake-up signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.

If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs ($T_{L1DevDrvResume}$) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

Table 471 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 471. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or suspended (Performing transactions)	2.5 µs
High speed	Not Idle or suspended (Performing transactions)	125 µs

58.14.41 OTG device status register (OTG_DSTS)

Address offset: 0x808

Reset value: 0x0000 0010

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSPSTS
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]:** Device line status

Indicates the current logic level USB data lines.
Bit [23]: Logic level of D+
Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]:** Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.
Due to erratic errors, the OTG controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]:** Enumerated speed

Indicates the speed at which the OTG controller has come up after speed detection through a chirp sequence.
00: High Speed using HS PHY
01: Full Speed using HS PHY
11: Full speed using embedded FS PHY
Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines

– When the application writes to the remote wake-up signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).

58.14.42 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

Address offset: 0x810

Reset value: 0x0000 0000

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	BNAM	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
		rw				rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **BNAM:** BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

58.14.43 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

Address offset: 0x814

Reset value: 0x0000 0000

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	BNAM	OUT PKT ERRM	Res.	B2B STUPM	STS PHSR XM	OTEPD M	STUPM	AHB ERRM	EPDM	XFRC M
	rw	rw	rw			rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK:** NYET interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BNAM:** BNA interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUPM:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **STSPHSRXM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** STUPM: SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

58.14.44 OTG device all endpoints interrupt register (OTG_DAINT)

Address offset: 0x818

Reset value: 0x0000 0000

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

58.14.45 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

Address offset: 0x81C

Reset value: 0x0000 0000

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 19 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

58.14.46 OTG device V_{BUS} discharge time register (OTG_DVBUSDIS)

Address offset: 0x0828

Reset value: 0x0000 17D7

This register specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VBUSDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VBUSDT[15:0]:** Device V_{BUS} discharge time

Specifies the V_{BUS} discharge time after V_{BUS} pulsing during SRP. This value equals:
V_{BUS} discharge time in PHY clocks / 1 024
Depending on V_{BUS} load, this value may need adjusting.

58.14.47 OTG device V_{BUS} pulsing time register (OTG_DVBUSPULSE)

Address offset: 0x082C

Reset value: 0x0000 05B8

This register specifies the V_{BUS} pulsing time during SRP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DVBUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DVBUSP[15:0]:** Device V_{BUS} pulsing time. This feature is only relevant to OTG1.3.

Specifies the V_{BUS} pulsing time during SRP. This value equals:
V_{BUS} pulsing time in PHY clocks / 1 024

58.14.48 OTG device threshold control register (OTG_DTHRCTL)

Address offset: 0x0830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN[8:0]									RXTHREN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXTHRLEN[8:0]									ISOTHREN	NONISOTHREN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **ARPEN:** Arbiter parking enable

This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.

Bit 26 Reserved, must be kept at reset value.

Bits 25:17 **RXTHRLEN[8:0]:** Receive threshold length

This field specifies the receive thresholding size in 32-bit words. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight 32-bit words. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 16 **RXTHREN:** Receive threshold enable

When this bit is set, the core enables thresholding in the receive direction.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:2 **TXTHRLEN[8:0]:** Transmit threshold length

This field specifies the transmit thresholding size in 32-bit words. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmitting on the USB. The threshold length has to be at least eight 32-bit words. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 1 **ISOTHREN:** ISO IN endpoint threshold enable

When this bit is set, the core enables thresholding for isochronous IN endpoints.

Bit 0 **NONISOTHREN:** Nonisochronous IN endpoints threshold enable

When this bit is set, the core enables thresholding for nonisochronous IN endpoints.

58.14.49 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

Address offset: 0x834

Reset value: 0x0000 0000

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INEPTXFEM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

58.14.50 OTG device IN endpoint x control register (OTG_DIEPCTLx)

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAKSTS	EONUM/DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USBAEP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS:** Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame
Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID
Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.
SEVNFRM: Set even frame
Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake
Applies to non-control, non-isochronous IN endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

58.14.51 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

Address offset: 0x908 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 707*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	BNA	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
		rc_w1		rc_w1		rc_w1	rc_w1	r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **BNA:** Buffer not available interrupt

The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done.

This bit is only valid when Scatter/Gather DMA mode is enabled.

Bit 8 **TXFIFOUDRN:** Transmit Fifo Underrun (TxfifoUndrn)

The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE:** IN token received when Tx FIFO is empty

Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

58.14.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

Address offset: 0x910

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		Res.	Res.	Res.
											rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]:** Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size
Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

58.14.53 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)

Address offset: 0x914 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address
This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

58.14.54 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

Address offset: 0x918 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0200

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INEPTFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available
Indicates the amount of free space available in the endpoint Tx FIFO.
Values are in terms of 32-bit words:
0x0: Endpoint Tx FIFO is full
0x1: 1 word available
0x2: 2 words available
0xn: n words available
Others: Reserved

58.14.55 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

Address offset: 0x910 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]:** Multi count
For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size
This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

58.14.56 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

Address offset: 0xB00

Reset value: 0x0000 8000

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to 2’b00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

00: 64 bytes

01: 32 bytes

10: 16 bytes

11: 8 bytes

58.14.57 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

Address offset: 0xB08 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 707*. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STPK TRX	NYET	NAK	BERR	Res.	Res.	BNA	OUT PKT ERR	Res.	B2B STUP	STSPH SRX	OTEP DIS	STUP	AHB ERR	EP DISD	XFRC
rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **STPKTRX:** Setup packet received

Applicable for control OUT endpoints in only in the Buffer DMA Mode. Set by the OTG, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per buffer. On receiving a setup packet, the OTG closes the buffer and disables the corresponding endpoint after SETUP_COMPLETE status is seen in the Rx FIFO. OTG puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN or OUT token after the SETUP packet for that particular endpoint. The application must then re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Because of the above behavior, OTG can receive any number of back to back setup packets and one buffer for every setup packet is used.

Bit 14 **NYET:** NYET interrupt

This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt

The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BNA:** Buffer not available interrupt

The core generates this interrupt when the descriptor accessed is not ready for the core to process, such as host busy or DMA done.

This bit is only valid when Scatter/Gather DMA mode is enabled.

Bit 8 **OUTPKTERR**: OUT packet error

This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. This interrupt is valid only when thresholding is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received

Applies to control OUT endpoint only.

This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 **STSPHSRX**: Status phase received for control write

This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled

Applies only to control OUT endpoints.

Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done

Applies to control OUT endpoint only.Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

58.14.58 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

Address offset: 0xB10

Reset value: 0x0000 0000

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–8.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.
	rw	rw										rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

58.14.59 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)

Address offset: 0xB14 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

58.14.60 OTG device OUT endpoint x control register (OTG_DOEPCTLx)

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID/ SODD FRM	SD0 PID/ SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM/ DPID
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

SODDFRM: Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk OUT endpoints only.

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

SEVNFRM: Set even frame

Applies to isochronous OUT endpoints only.

Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Applies to control endpoints only (access type is rs).

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN and OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

DPID: Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

58.14.61 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

Address offset: 0xB10 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID/ STUPCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID[1:0]:** Received data PID
Applies to isochronous OUT endpoints only.
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT[1:0]: SETUP packet count
Applies to control OUT endpoints only.
This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size
This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

58.14.62 OTG power and clock gating control register (OTG_PCGCCTL)

Address offset: 0xE00

Reset value: 0x200B 8000

This register is available in host and device modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SUSP:** Deep Sleep

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP:** PHY in Sleep

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG:** Enable sleep clock gating

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wake-up logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

58.14.63 OTG register map

The table below gives the USB OTG register map and reset values.

Table 472. OTG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_GOTGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS	Res.	Res.	Res.	EHEN	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN	SRQ	SRQSCS
	Reset value											0	0	0	0	0	1				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	OTG_GOTGINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBCDNE	ADTOCHG	HNGDET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HNSSCHG	SRSSCHG	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
	Reset value													0	0	0							-	0	0						0		
0x008	OTG_GAHBCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	DMAEN	HBSTLEN				GINTMSK
	Reset value																								0	0		0	0	0	0	0	0
0x00C	OTG_GUSBCFG	Res.	FDMOD	FHMOD	Res.	Res.	Res.	Res.	Res.	Res.	TSDPS	Res.	Res.	Res.	Res.	Res.	Res.	PHYLPC	Res.	TRDT				HNPCA	SRPCAP	Res.	PHYSEL	Res.	Res.	Res.	TOCAL		
	Reset value		0	0														0		0	1	0	1	0	0		1				0	0	0
0x010	OTG_GRSTCTL	AHBIDL	DMAREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFNUM					TXFFLSH	RXFFLSH	Res.	FCRST	PSRST	CSRST
	Reset value	1	0																				0	0	0	0	0	0	0		0	0	0
0x014	OTG_GINTSTS	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	OTG_ GRXSTSR (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSR (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_ GRXSTSP (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSP (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_ GRXFSIZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	OTG_ HNPTXFSIZ/ OTG_ DIEPTXF0	NPTXFD/TX0FD																NPTXFSA/TX0FSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	OTG_ HNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	OTG_ GCCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDEN	VBDEN	SDEN	PDEN	Res.	BCDEN	PWRDWN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	Res.
	Reset value										0	0	0	0		0	0													X	X	X	
0x03C	OTG_CID	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	OTG_ GLPMCFG	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT			LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS				L1SSEN	REMWAKE	BESL				LPMACK	LPMEN
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	OTG_ HPTXFSIZ	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x104	OTG_ DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	OTG_ DIEPTXF2	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
.	.																.																
.	.																.																
0x120	OTG_ DIEPTXF7	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_ HCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLSPCS	
	Reset value																														0	0	0
0x404	OTG_ HFIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTRL	FRIVL															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_ HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_ HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x414	OTG_ HAINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_ HAINTMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x41C	OTG_ HFLBADDR	HFLBADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_ HPRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD		PTCTL				PPWR	PLSTS		Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	Reset value														0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x500	OTG_ HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x504	OTG_ HCSPLT0	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_HCINTMSK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	OTG_HCTSIZ0	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x510	OTG_HCTSIZSG0	DOPNG	PID		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTD								SCHED_INFO							
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	OTG_HCDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	OTG_HCDMASG0	DMASG[31:3]																													Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x51C	OTG_HCDMAB0	HCDMAB																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	OTG_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x524	OTG_HCSPLT1	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	OTG_HCINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x528	OTG_HCINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x52C	OTG_ HCINTMSK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x530	OTG_ HCTSIZ1	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	OTG_ HCTSIZSG1	DOPNG	PID		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTD								SCHED_INFO							
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x534	OTG_ HCDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x534	OTG_ HCDMASG1	DMASG[31:3]																													Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-
0x53C	OTG_ HCDMAB1	HCDMAB																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	. . . .																															
0x660	OTG_ HCCHAR11	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x664	OTG_ HCSPLT11	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	OTG_ HCINT11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x66C	OTG_ HCINTMSK11	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x670	OTG_ HCTSIZ11	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x670	OTG_ HCTSIZSG11	DOPNG	PID		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTD								SCHED_INFO							
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x674	OTG_ HCDMA11	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x674	OTG_ HCDMASG11	DMASG[31:3]																													Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-
0x67C	OTG_ HCDMAB11	HCDMAB																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	. . . .																															
0x6E0	OTG_ HCCHAR15	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E4	OTG_ HCSPLT15	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E8	OTG_ HCINT15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x6EC	OTG_ HCINTMSK15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x6F0	OTG_ HCTSIZ15	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F0	OTG_ HCTSIZSG15	DOPNG	PID		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTD								SCHED_INFO							
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F4	OTG_ HCDMA15	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6F4	OTG_HCDMASG15	DMASG[31:3]																													Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-
0x6FC	OTG_HCDMAB15	HCDMAB																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	OTG_DCFG	Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL		DESCDMA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRATIM	XCVRDLY	Res.	PFIVL		DAD							Res.	NZLSOHSK	DSPD	
	Reset value							0	0	0								0	0		0	0	0	0	0	0	0	0	0		0	0	0
0x804	OTG_DCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	ENCONTONBN	Res.	Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value														0	0						0	0	0	0	0	0	0	0	0	0	1	0
0x808	OTG_DSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS		FNSOF														Res.	Res.	Res.	Res.	EERR	ENUMSPD		SUSPSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	OTG_DIEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	BNAM	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0				0	0		0	0	0	0	0	0	0
0x814	OTG_DOEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYETMSK	NAKMSK	BERRM	Res.	Res.	BNAM	OUTPKTERRM	Res.	B2BSTUPM	Res.	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																		0	0	0			0	0		0		0	0	0	0	0
0x818	OTG_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	OTG_DVBUSDIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	OTG_DVBUSPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVBUSP															
	Reset value																	0	0	0	0	0	1	0	1	1	0	1	1	1	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x830	OTG_DTHRCTL	Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN									RXTHREN	Res.	Res.	Res.	Res.	TXTHRLEN										ISOTHREN	NONISOTHREN
	Reset value					0											0					0	0	0	0	0	0	0	0	0	0	0	0
0x834	OTG_DIEPEMPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x900	OTG_DIEPCTL0	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_DIEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	BNA	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0		0	0	1	0	0	0	0	0	0	0
0x910	OTG_DIEPTSIZ0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x914	OTG_DIEPDMA	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x918	OTG_DTXFSTS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	OTG_DIEPCTL1	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x928	OTG_DIEPINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	BNA	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0		0	0	1	0	0	0	0	0	0	0
0x930	OTG_DIEPTSIZ1	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x938	OTG_DTXFSTS1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x940	OTG_DIEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
...	...	...																															
...	...	...																															
...	...	...																															
...	...	...																															
0x9E0	OTG_DIEPCTL7	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
...	...	...																															
0x9E8	OTG_DIEPINT7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	BNA	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0		0	0	1	0	0	0	0	0	0	0
...	...	...																															
0x9F0	OTG_DIEPTSIZ7	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
...	...	...																															
0x9F8	OTG_ DTXFSTS7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0xB00	OTG_ DOEPCTL0	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB08	OTG_ DOEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	BNA	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0			0	0		0	0	0	0	0	0	0
0xB10	OTG_ DOEPTSIZ0	Res.	STUP CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB14	OTG_ DOEPDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB20	OTG_ DOEPCTL1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB28	OTG_ DOEPINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	BNA	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0			0	0		0	0	0	0	0	0	0
0xB30	OTG_ DOEPTSIZ1	Res.	RXDPID/ STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB34	OTG_ DOEPDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...																															

Table 472. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC00	OTG_ DOEPCTL8	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xC08	OTG_ DOEPINT8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	BNA	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0		-	0	0		0	0	0	0	0	0	0
0xC10	OTG_ DOEPTSIZ8	Res.	RXDPID/ STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC14	OTG_ DOEPDMA8	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_ PCGCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATEHCLK	STPPCLK
	Reset value																									0	0	0	0			0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

58.15 OTG programming model

58.15.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level
2. Program the following fields in the OTG_GUSBCFG register:
   - HNP capable bit
   - SRP capable bit
   - OTG timeout calibration field
   - USB turnaround time field
3. The software must unmask the following bits in the OTG_GINTMSK register:

   OTG interrupt mask

   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG controller is operating in host or device mode.

58.15.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. This drives V_{BUS} on the USB.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

58.15.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   – DESCDMA
   – Device speed
   – Non-zero-length status OUT handshake
   – Periodic Frame Interval
2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable thresholding.
3. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
4. Program the OTG_GINTMSK register to unmask the following interrupts:
   – USB reset
   – Enumeration done
   – Early suspend
   – USB suspend
   – SOF
5. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
6. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 3110*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

58.15.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (OTG_HCDMAx register in host mode and OTG_DIEPDMAx/OTG_DOEPDMAx register in peripheral mode) to access the data buffers.

Scatter/Gather DMA mode

In Scatter/Gather DMA mode, the core implements a true scatter/gather memory distribution in which data buffers are scattered over the system memory. However, the descriptors themselves are continuous. Each channel memory structure is implemented as a contiguous list of descriptors; each descriptor points to a data buffer of predefined size. In addition to the buffer pointer (a 32-bit word), the descriptor also has a status quadlet (32-bit word). When the list is implemented as a ring buffer, the list processor switches to the first element of the list when it encounters last bit. All channels (control, bulk, interrupt, and isochronous) implement these structures in memory. When Scatter/Gather DMA is enabled in device mode, OTG_DIEPDMAx and OTG_DOEPDMAx registers are used to access the base descriptor.

58.15.5 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).
7. Program the selected channels in the OTG_HCSPLTx register(s) with the hub and port addresses (split transactions only).
8. Program the selected channels in the OTG_HCDMAx register(s) with the buffer start address (DMA transactions only).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG host checks for space to write the disable

request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in OTG_HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When an XFRC interrupt in OTG_HCINTx is received during a non periodic IN transfer or high-bandwidth interrupt IN transfer
3. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).
4. When the application aborts a transfer before normal completion.

Ping protocol

When the OTG host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints. The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the OTG host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO).This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in OTG_HCTSIZx before enabling the channel or by just writing the OTG_HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the OTG host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to set the DOPING bit in OTG_HCTSIZx for a NAK/NYET response in case of bulk/control OUT. The OTG host automatically sets the DOPING bit in OTG_HCTSIZx, and issues the ping tokens for bulk/control OUT. The OTG host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

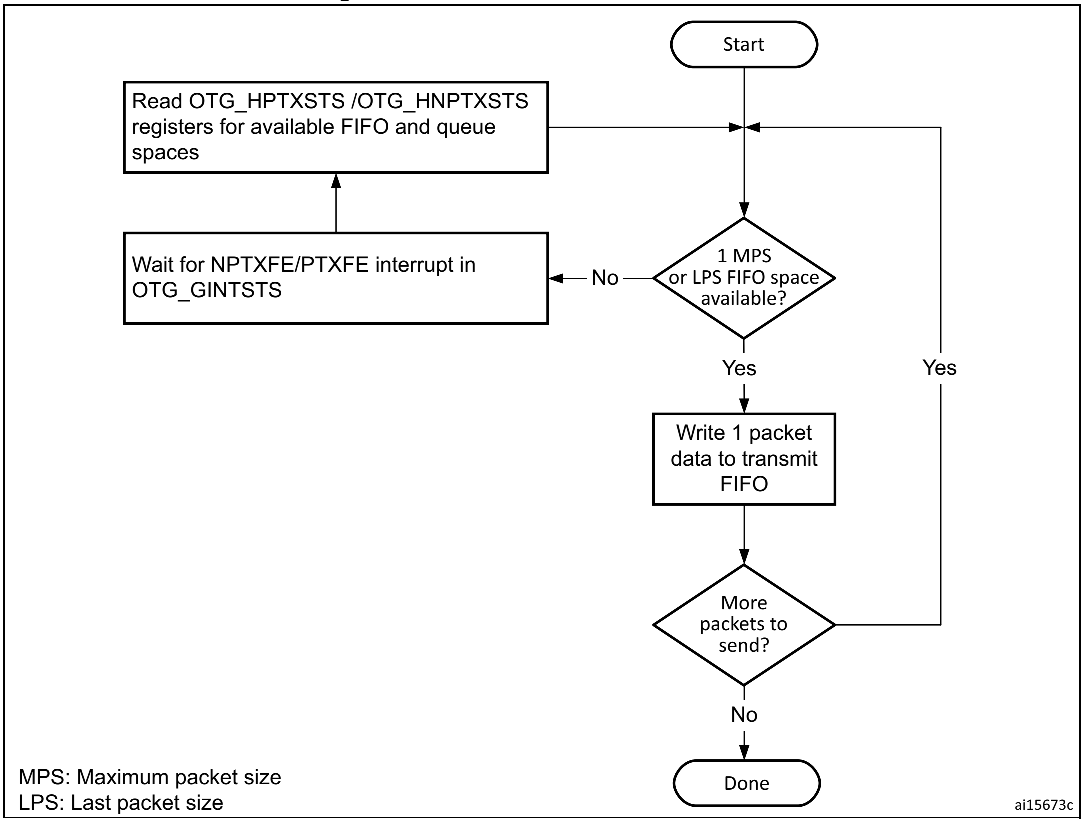
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG host determines the actual packet size based on the programmed maximum packet size and transfer size.

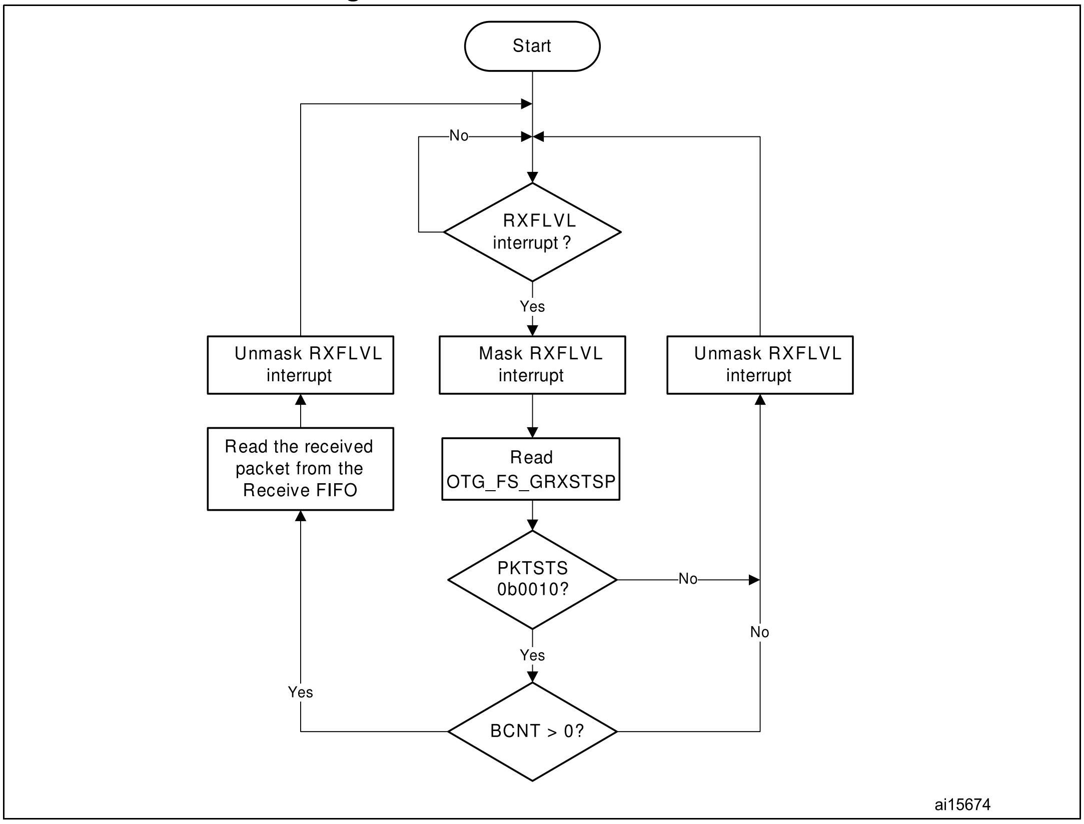
Figure 708. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 709. Receive FIFO read task



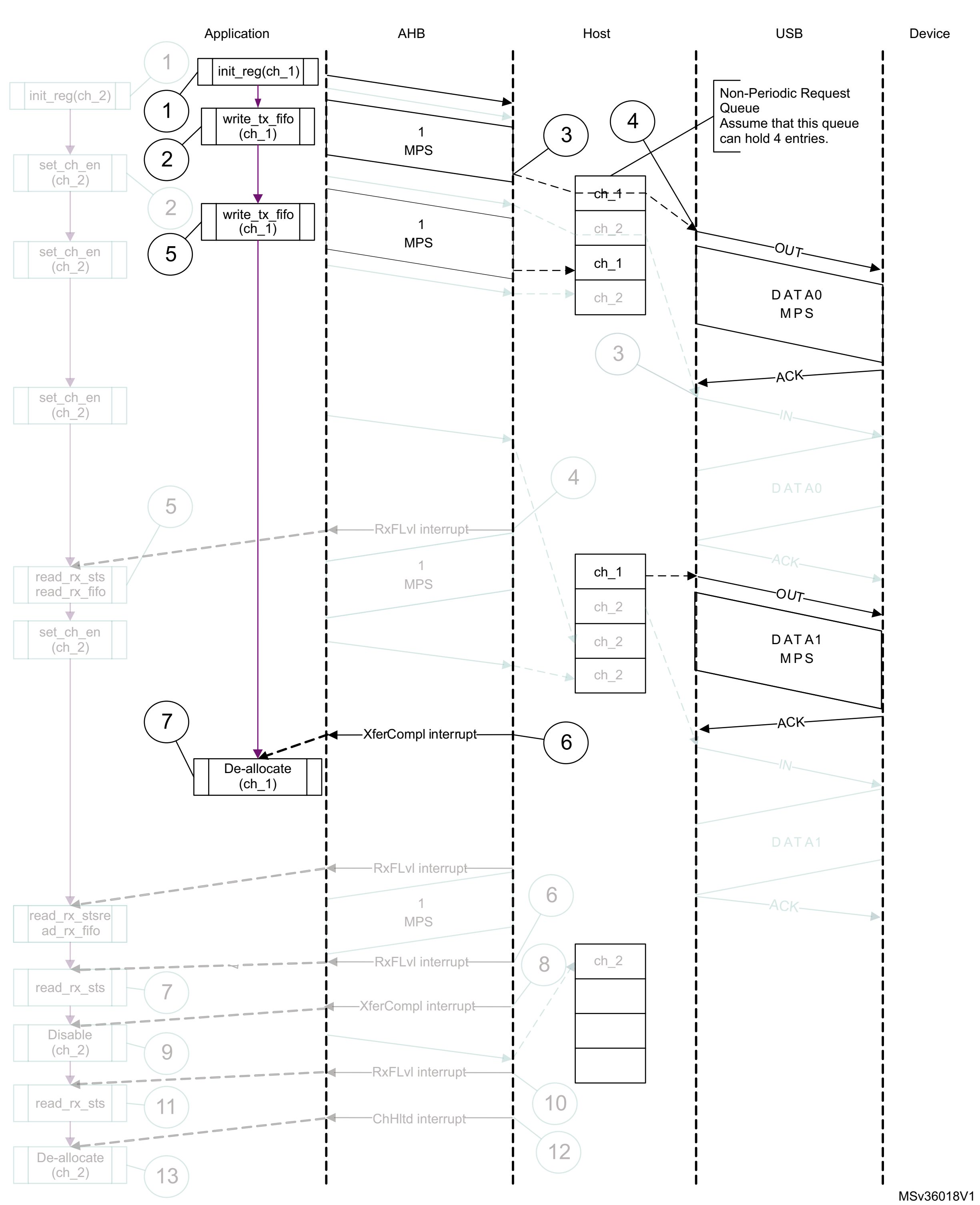
- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 710*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:
  - The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
  - The non-periodic transmit FIFO can hold two packets (1 Kbyte for HS).
  - The non-periodic request queue depth = 4.
- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 710. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
```

```
        else if (ACK)
          {
          Reset Error Count
          Mask ACK
          }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
  {
  Reset Error Count
  Unmask CHH
  Disable Channel
  Reset Error Count
  Mask ACK
  }
else if (TXERR or BBERR or STALL)
  {
  Unmask CHH
  Disable Channel
  if (TXERR)
    {
    Increment Error Count
    Unmask ACK
    }
  }
else if (CHH)
  {
  Mask CHH
  if (Transfer Done or (Error_count == 3))
    {
    De-allocate Channel
    }
  else
    {
    Re-initialize Channel
    }
  }
else if (ACK)
  {
  Reset Error Count
  Mask ACK
  }
```

```
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

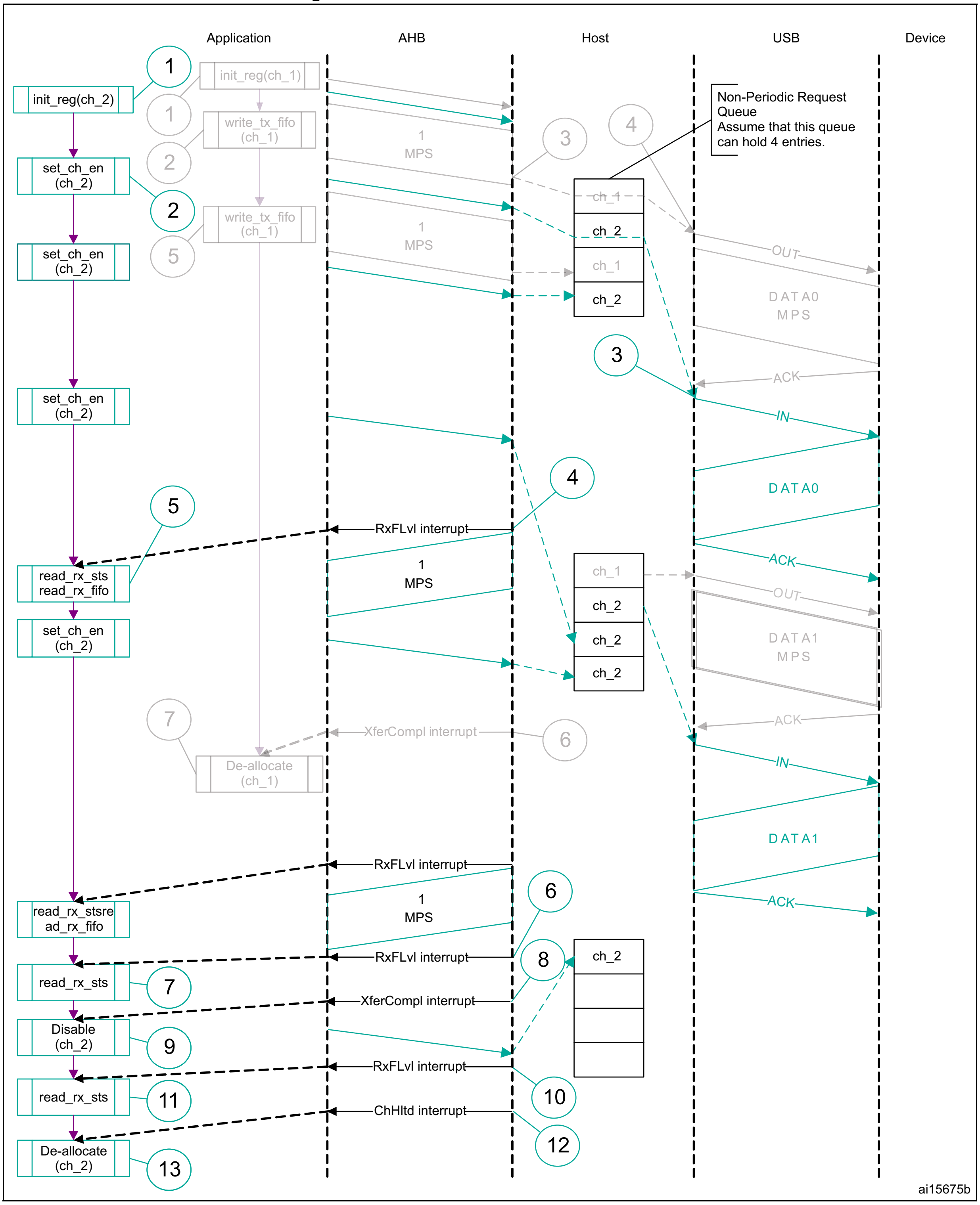
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 711*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (520 bytes for HS).

  – The non-periodic request queue depth = 4.

Figure 711. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.

The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in

OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 712*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

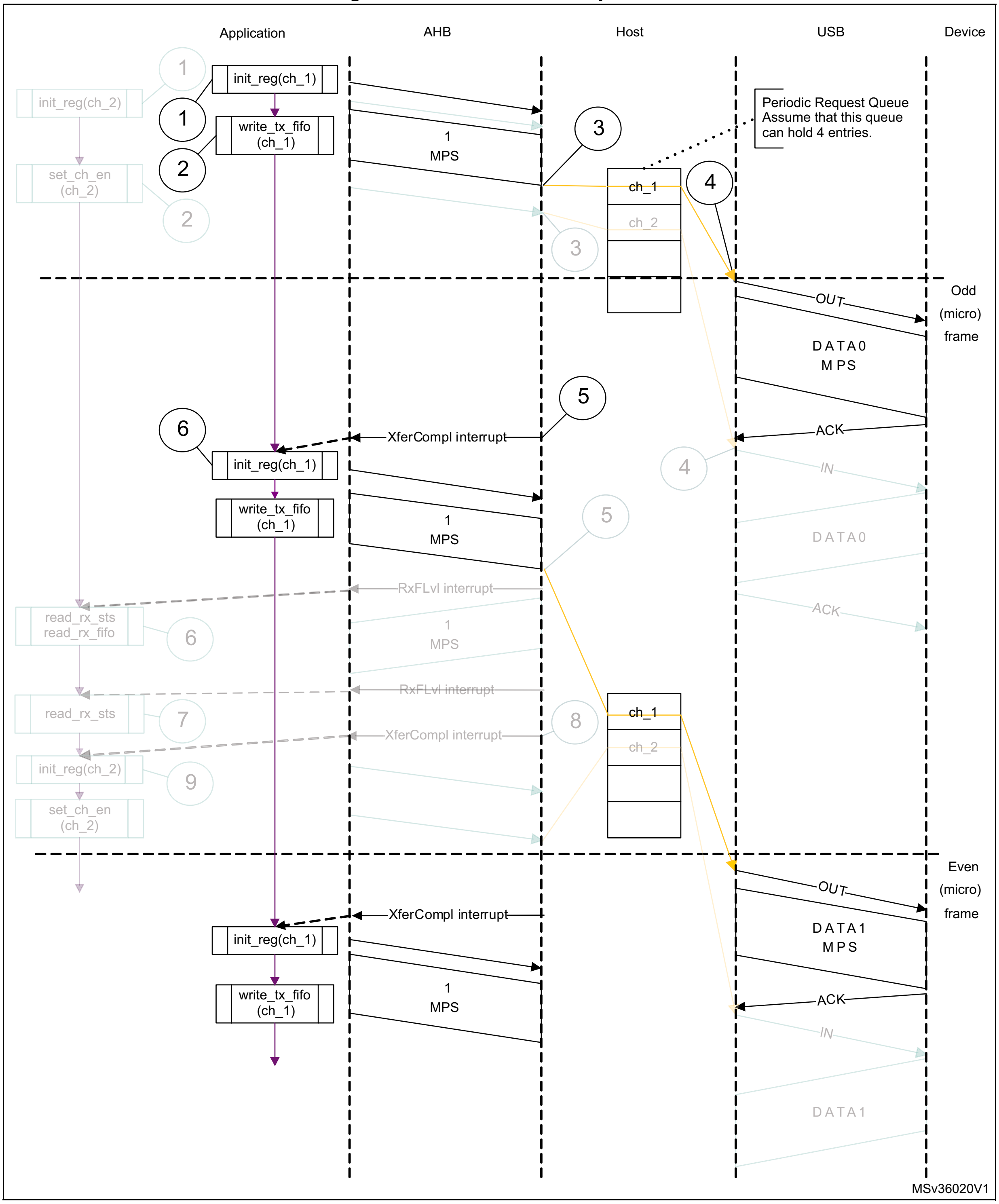
  – The periodic transmit FIFO can hold one packet (1 Kbyte)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG host writes an entry to the periodic request queue.
4. The OTG host attempts to send an OUT token in the next (odd) frame.
5. The OTG host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 712. Normal interrupt OUT



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

a) Interrupt OUT

`Unmask (NAK/TXERR/STALL/XFRC/FRMOR)`

```
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

```
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the
transmit FIFO space.
Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
```

```
    if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
      else
        Re-initialize Channel (in next b_interval - 1 /Frame)
      }
    }
else
    if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

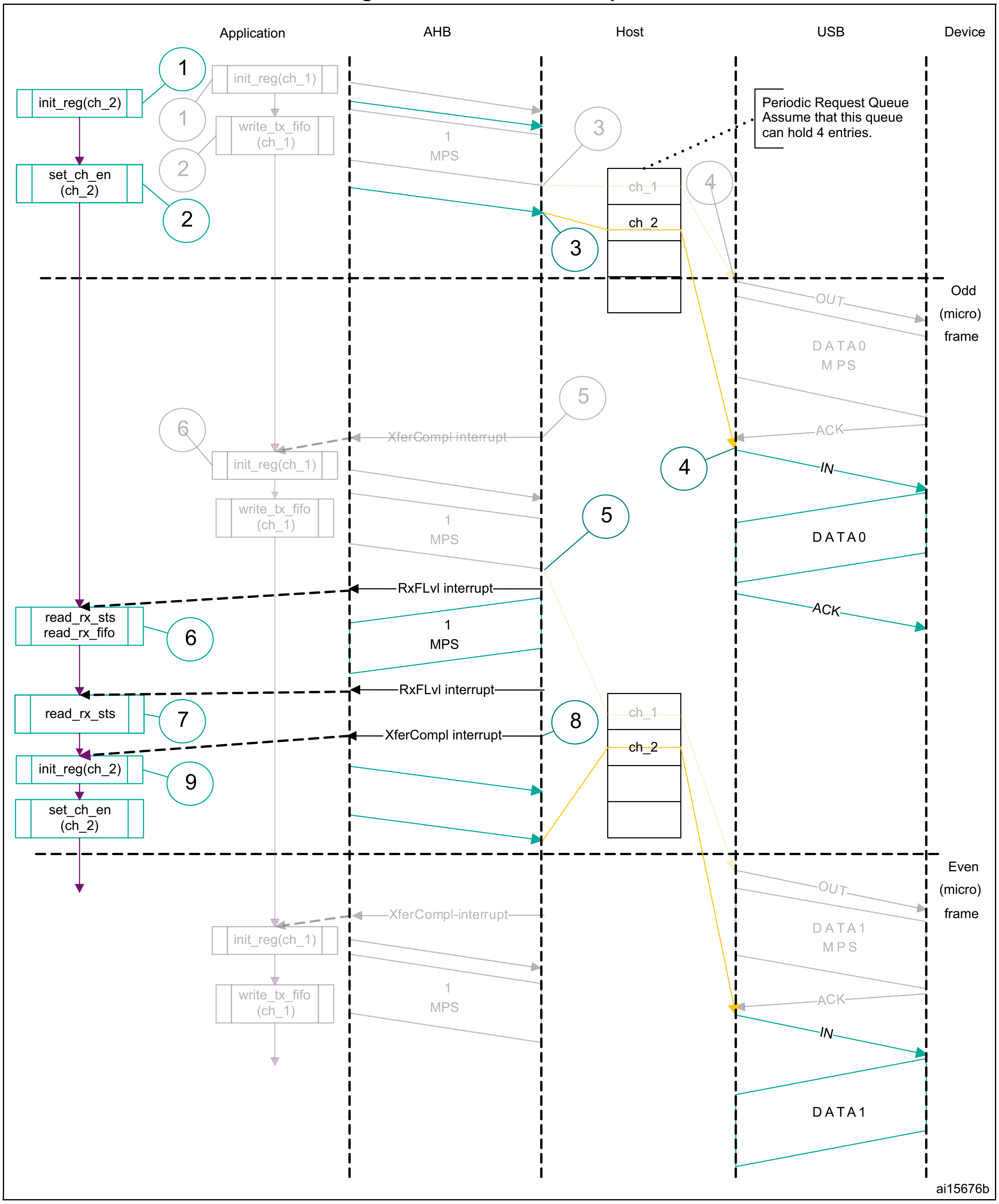
  The assumptions are:
  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).
  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).
  – Periodic request queue depth = 4.
- **Normal interrupt IN operation**

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 713. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 714*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum packet size), starting with an odd frame. (transfer size = 1 024 bytes).

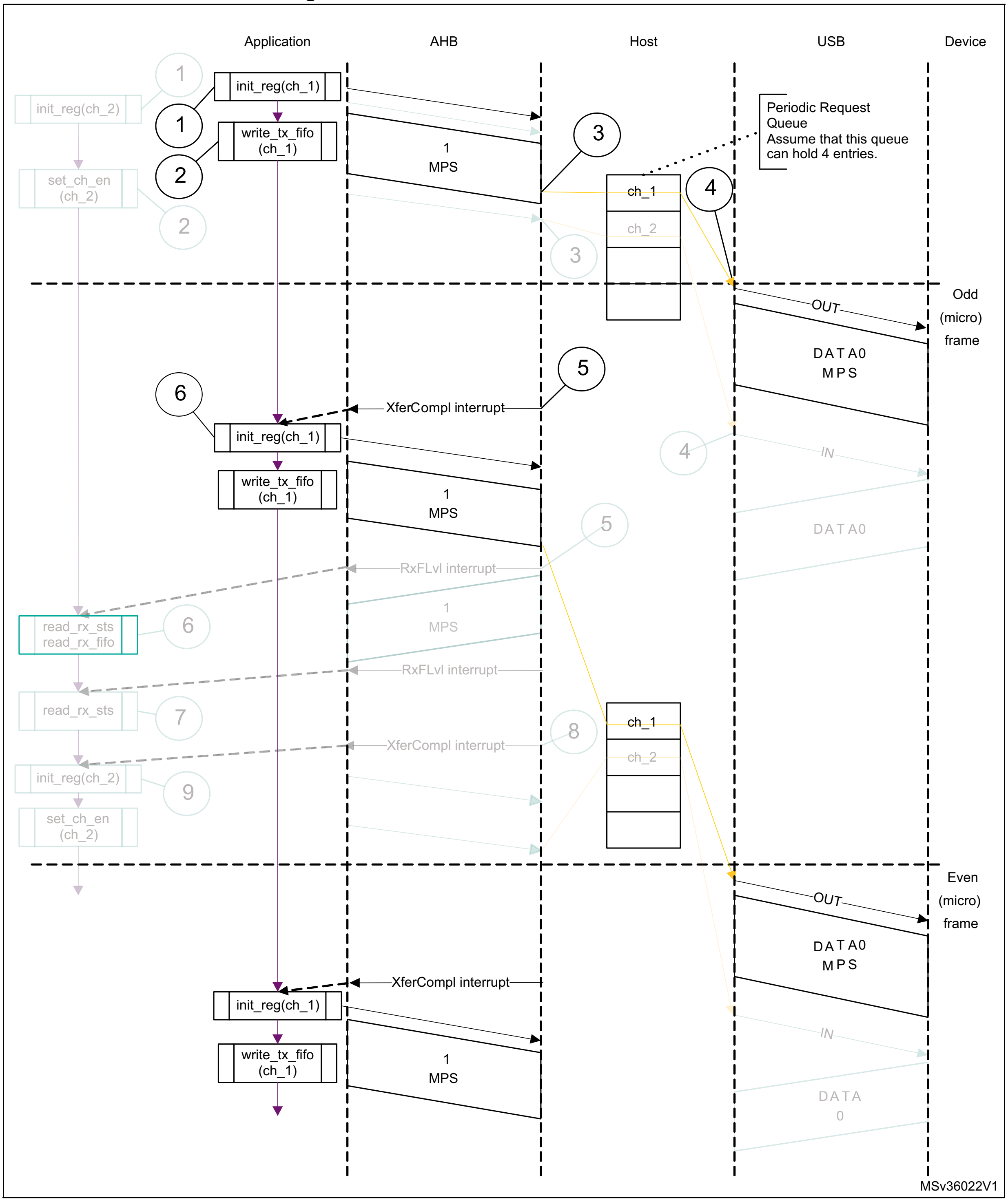
– The periodic transmit FIFO can hold one packet (1 Kbyte).

– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG host writes an entry to the periodic request queue.
4. The OTG host attempts to send the OUT token in the next frame (odd).
5. The OTG host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 714. Isochronous OUT transactions



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
```

```
      {
      De-allocate Channel
      }
   else
      if (FRMOR)
         {
         Unmask CHH
         Disable Channel
         }
      else
      if (CHH)
         {
         Mask CHH
         De-allocate Channel
         }
   Code sample: Isochronous IN
   Unmask (TXERR/XFRC/FRMOR/BBERR)
   if (XFRC or FRMOR)
      {
      if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
         {
         Reset Error Count
         De-allocate Channel
         }
      else
         {
         Unmask CHH
         Disable Channel
         }
      }
      else
         if (TXERR or BBERR)
            {
            Increment Error Count
            Unmask CHH
            Disable Channel
            }
         else
            if (CHH)
               {
               Mask CHH
               if (Transfer Done or (Error_count == 3))
                  {
                  De-allocate Channel
                  }
```

```
else
   {
   Re-initialize Channel
   }
}
```

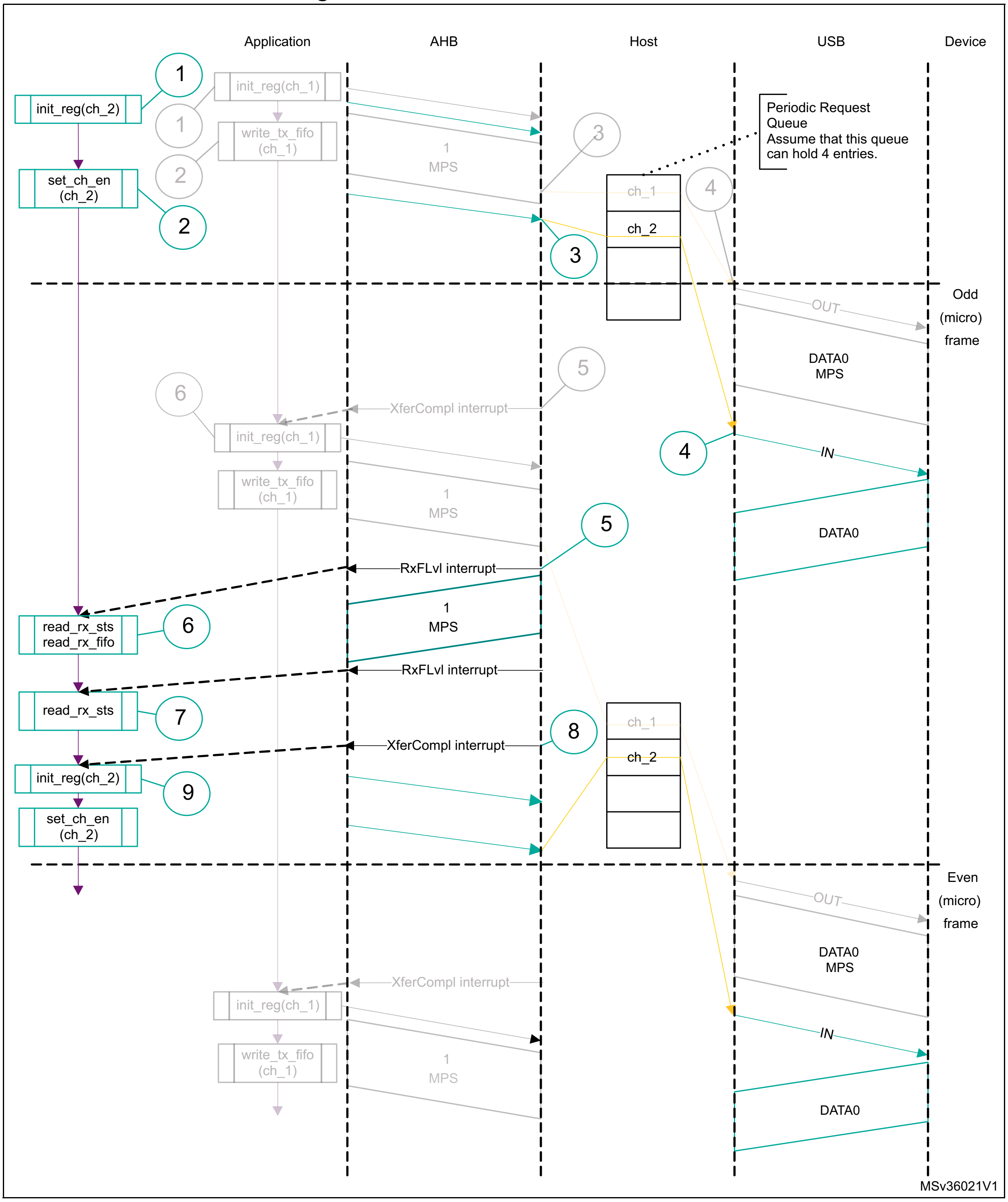
- **Isochronous IN transactions**

  The assumptions are:
  - The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).
  - The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).
  - Periodic request queue depth = 4.

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 715. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core’s periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

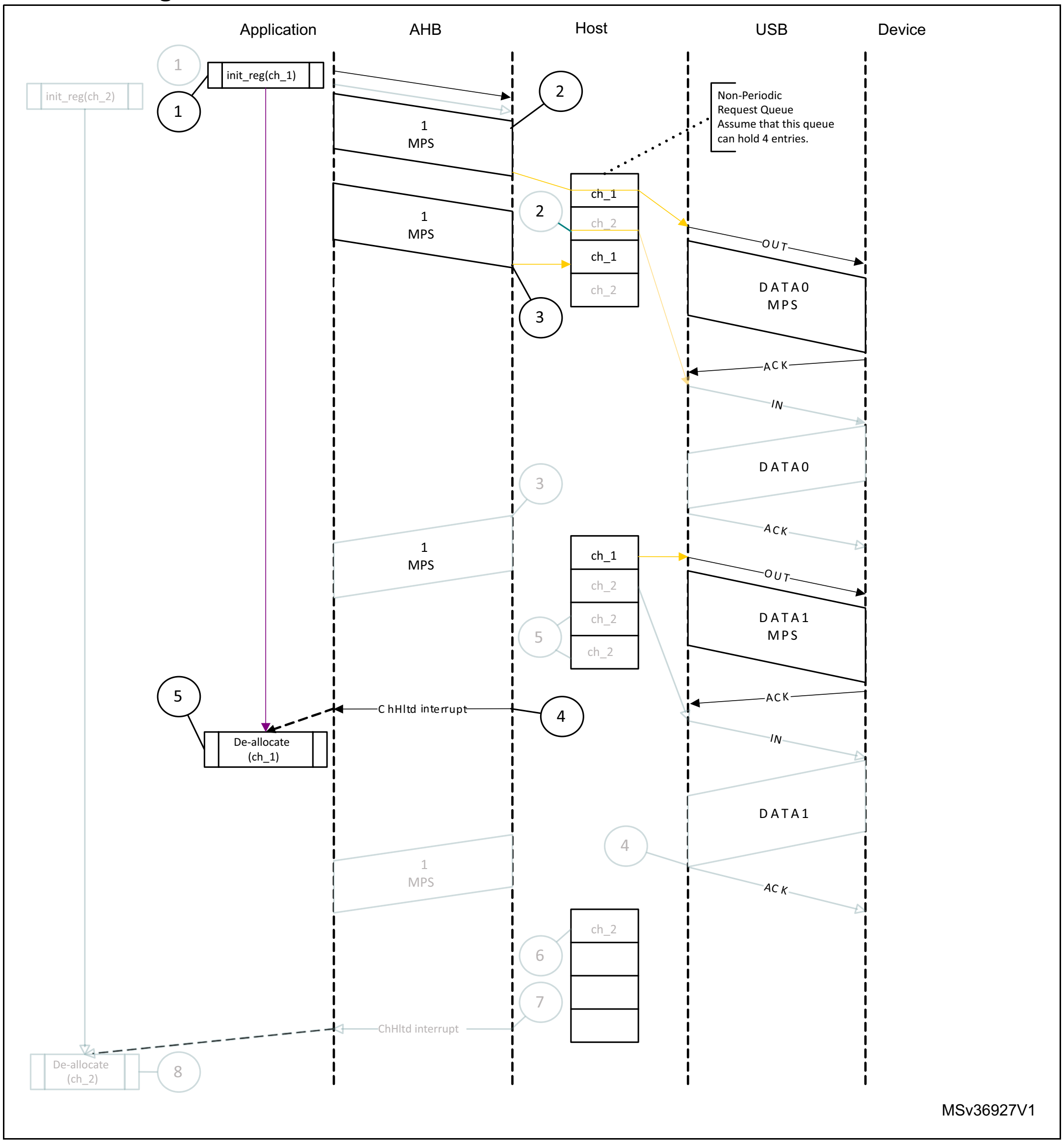
  When OTG controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable channel 1 as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG host uses the programmed DMA address to fetch the packet.
3. After fetching the last 32-bit word of the second (last) packet, the OTG host masks channel 1 internally for further arbitration.
4. The OTG host generates a CHH interrupt as soon as the last packet is sent.
5. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 716. Normal bulk/control OUT/SETUP transactions - DMA



- **NAK and NYET handling with internal DMA:**

1. The OTG host sends a bulk OUT transaction.
2. The device responds with NAK or NYET.
3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.
4. The core automatically issues a ping token.
5. When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.

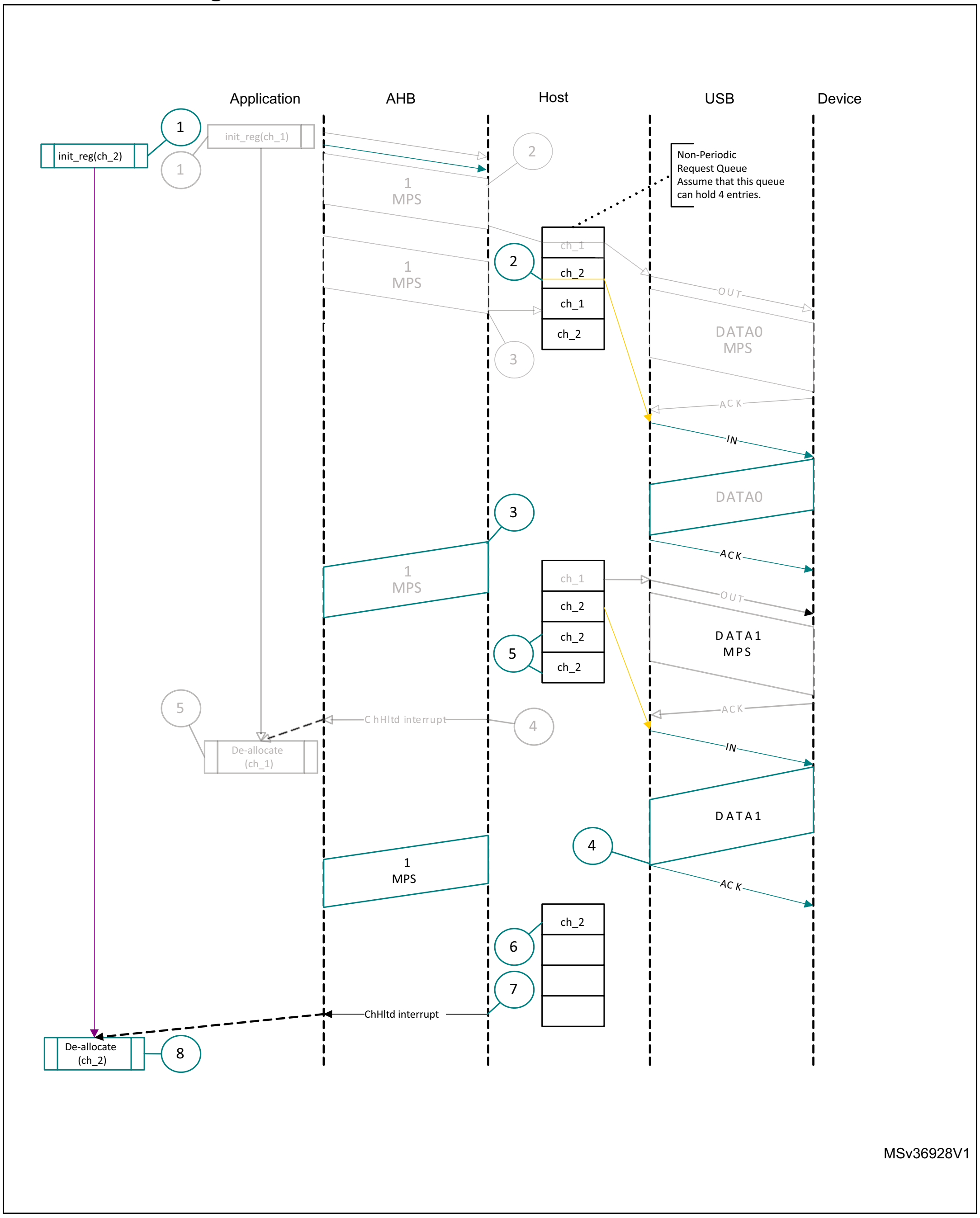
The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable the used channel (channel x) as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).
3. The OTG host starts writing the received data to the system memory as soon as the last byte is received with no errors.
4. When the last packet is received, the OTG host sets an internal flag to remove any extra IN requests from the request queue.
5. The OTG host flushes the extra requests.
6. The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.
7. The OTG host generates the CHH interrupt as soon as the disable request comes to the top of the queue.
8. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 717. Normal bulk/control IN transaction - DMA

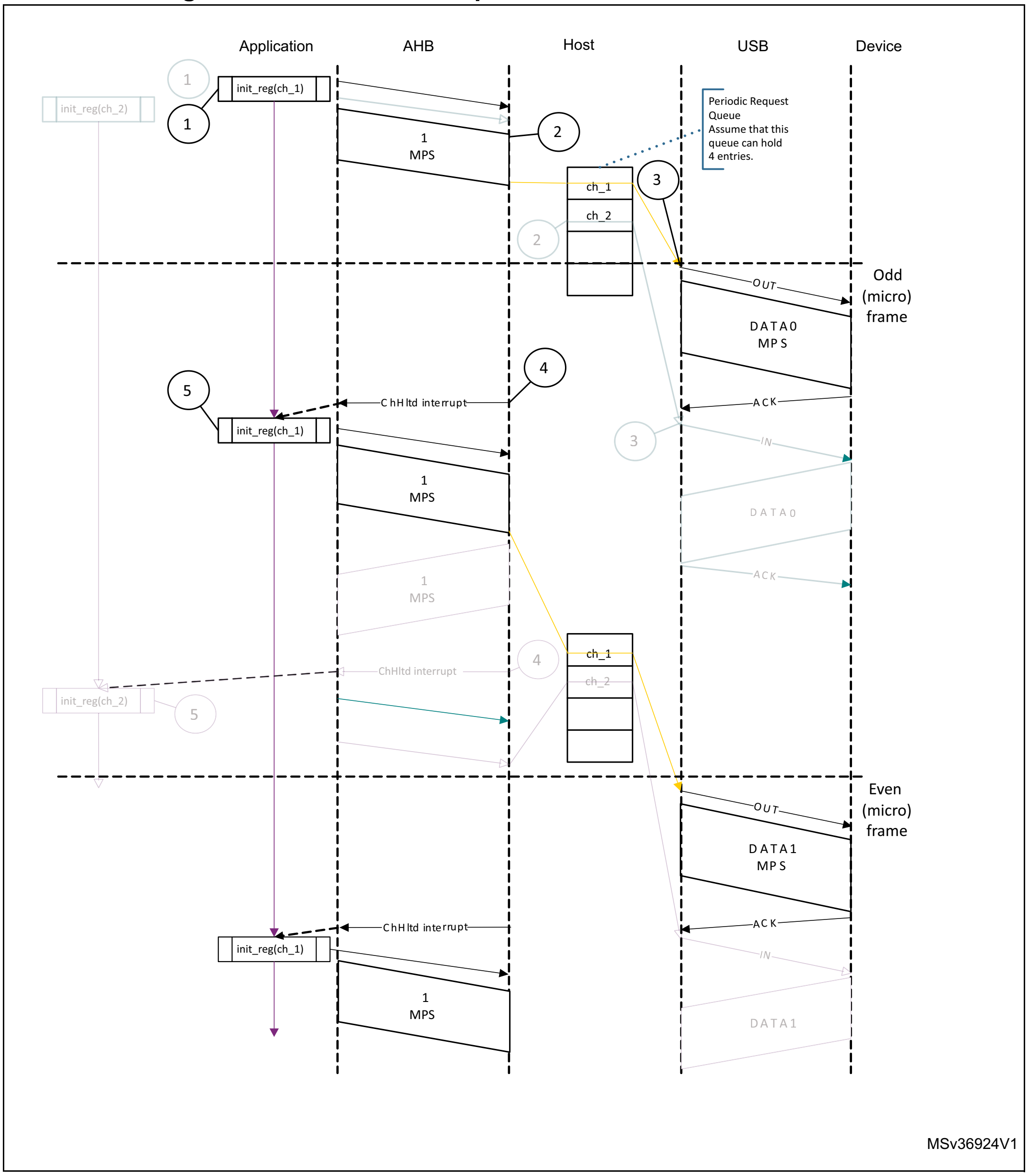


- **Interrupt OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth transfers, the OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The OTG host attempts to send the OUT token at the beginning of the next odd frame/micro-frame.

4. After successfully transmitting the packet, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 718. Normal interrupt OUT transactions - DMA mode



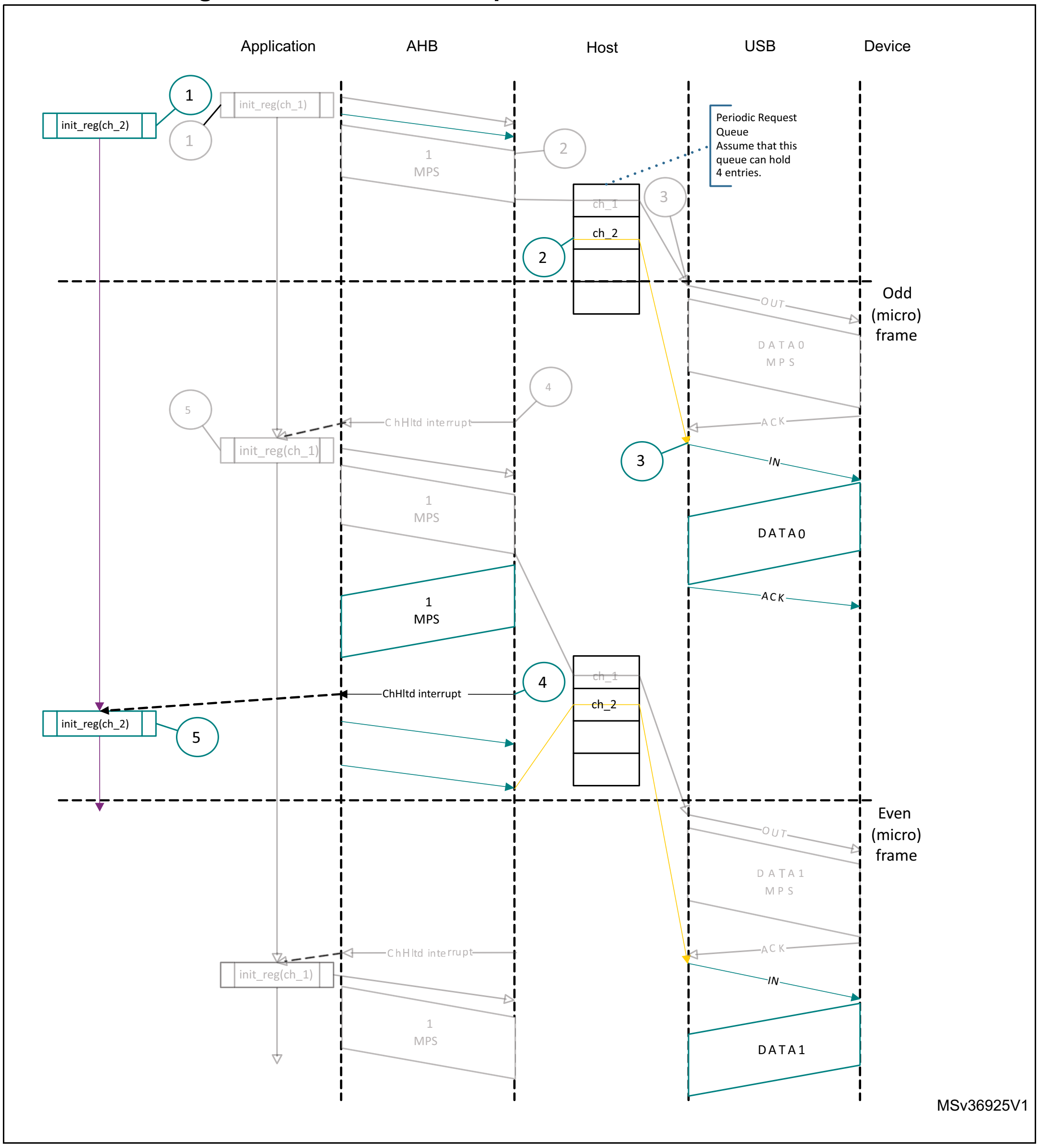
- **Interrupt IN transactions in DMA mode**

  The sequence of operations (channelx) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG host writes consecutive writes up to MC times.
3. The OTG host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 719. Normal interrupt IN transactions - DMA mode



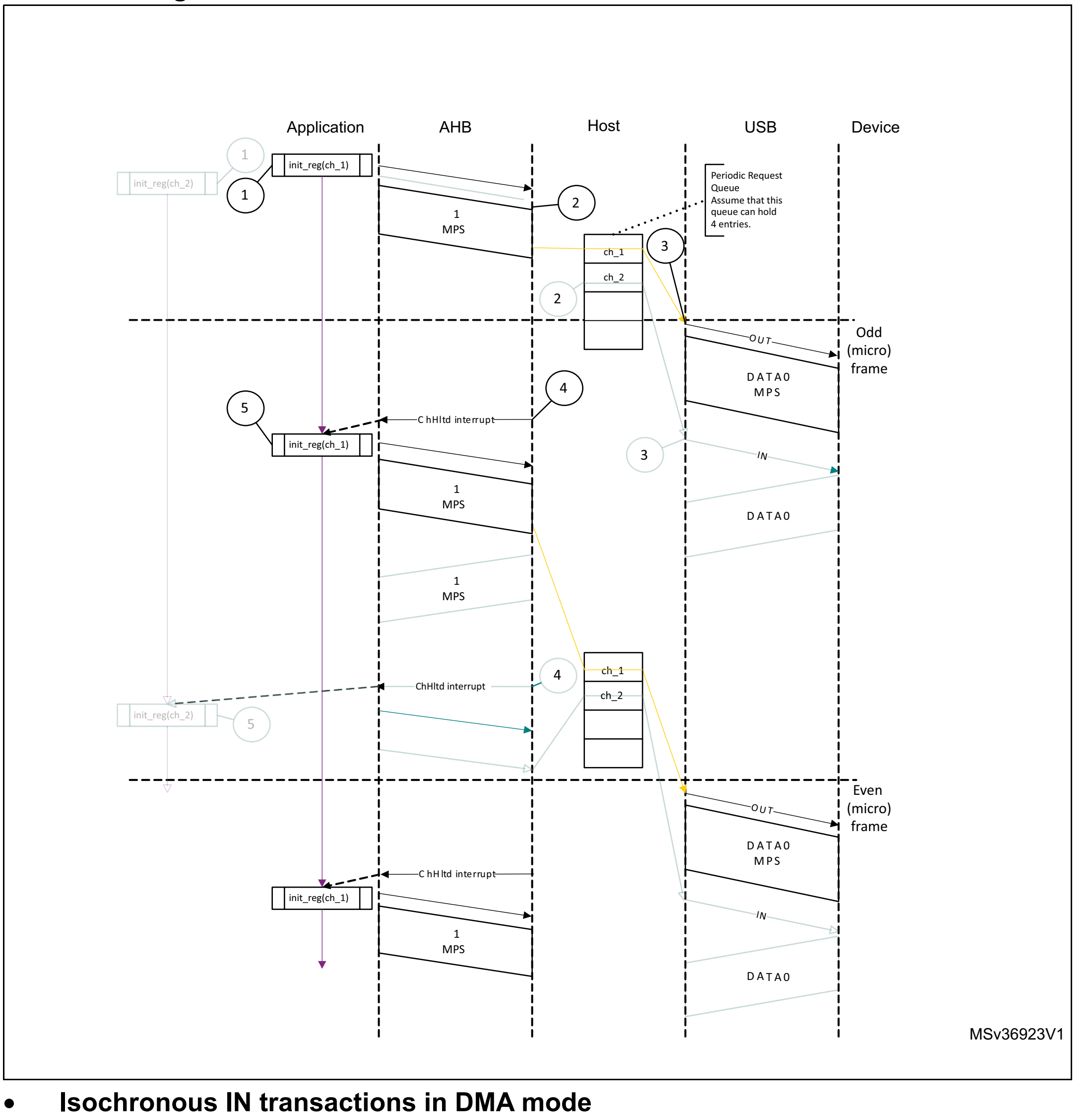
- **Isochronous OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth

transfers, the OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

3. The OTG host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.
4. After successfully transmitting the packet, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 720. Normal isochronous OUT transaction - DMA mode



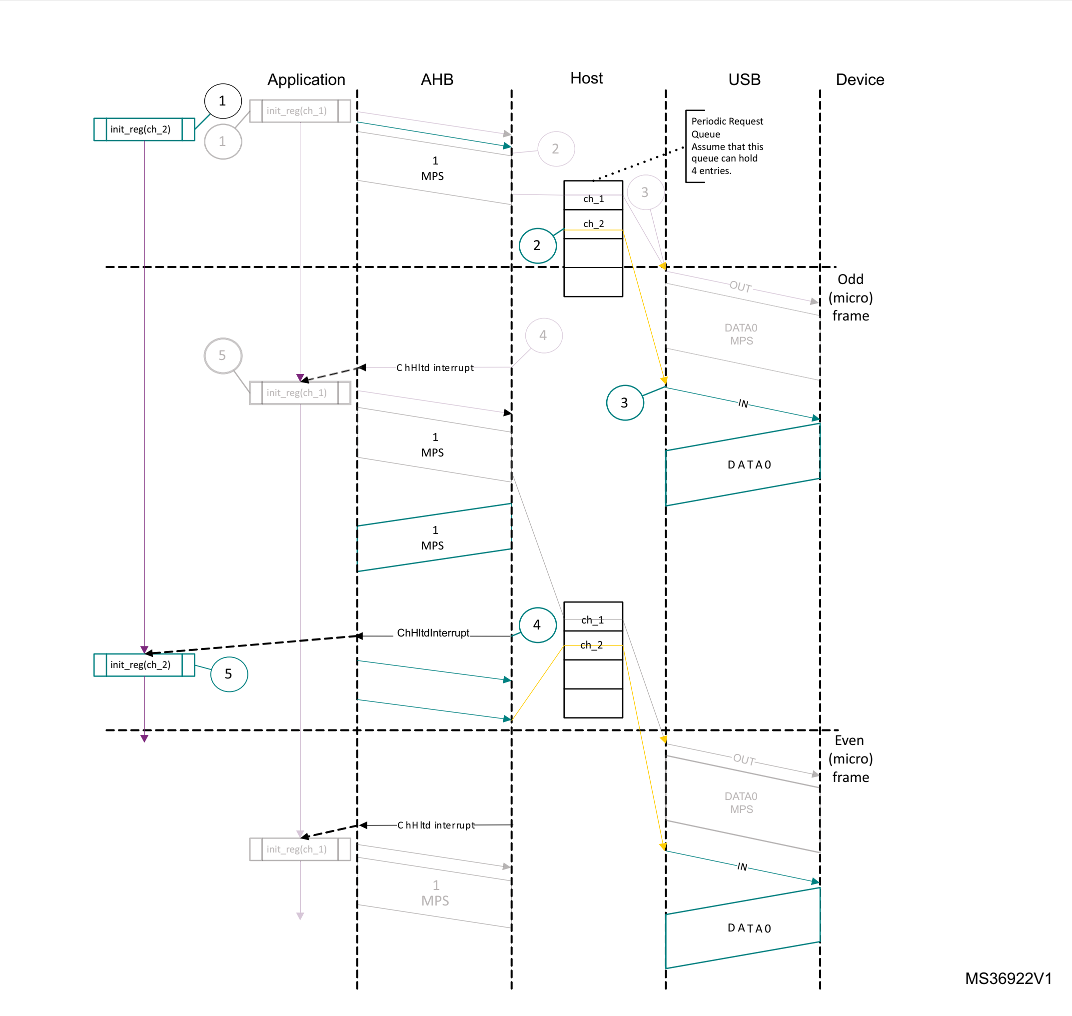
- **Isochronous IN transactions in DMA mode**

The sequence of operations ((channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG host performs consecutive write operations up to MC times.
3. The OTG host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 721. Normal isochronous IN transactions - DMA mode



- **Bulk and control OUT/SETUP split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch.
3. After successfully transmitting start split, the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
5. After successfully transmitting complete split, the OTG host generates the CHH interrupt.

6. In response to the CHH interrupt, de-allocate the channel.

- **Bulk/control IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG host masks the channel x internally for the arbitration after writing the request.
3. As soon as the IN token is transmitted, the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 and re-enable the channel to send the complete split token. This unmasks channel x for arbitration.
5. The OTG host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.
6. The OTG host starts writing the packet to the system memory after receiving the packet successfully.
7. As soon as the received packet is written to the system memory, the OTG host generates a CHH interrupt.
8. In response to the CHH interrupt, de-allocate the channel.

- **Interrupt OUT split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel 1 for start split as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. The OTG host starts reading the packet.
3. The OTG host attempts to send the start split transaction.
4. After successfully transmitting the start split, the OTG host generates the CHH interrupt.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
6. After successfully completing the complete split transaction, the OTG host generates the CHH interrupt.
7. In response to CHH interrupt, de-allocate the channel.

- **Interrupt IN split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.

6. As soon as the packet is received successfully, the OTG host starts writing the data to the system memory.
7. The OTG host generates the CHH interrupt after transferring the received data to the system memory.
8. In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- **Isochronous OUT split transactions in DMA mode**

  The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split (begin) as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1. Program the MPS field.
2. The OTG host starts reading the packet.
3. After successfully transmitting the start split (begin), the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, reinitialize the registers to send the start split (end).
5. After successfully transmitting the start split (end), the OTG host generates a CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

- **Isochronous IN split transactions in DMA mode**

  The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.
6. As soon as the packet is received successfully, the OTG host starts writing the data to the system memory.

   The OTG host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.

58.15.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   - SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   - INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   - OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   - STUPM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DIEPMSK
   - TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   - Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   - Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. For USB OTG in DMA mode, the OTG_DOEPDMA0 register should have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. For USB OTG in DMA mode, program the OTG_DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

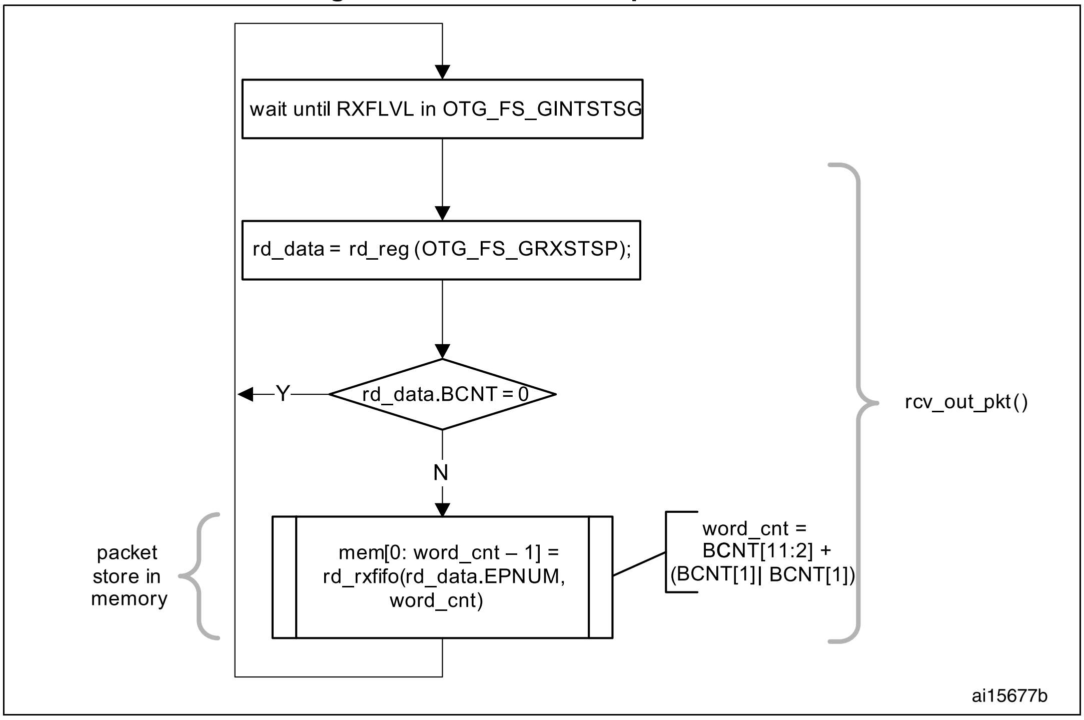
- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
      PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0), DPID = (0b00).
      These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
      PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num, DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
      PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).
      These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
      PKTSTS = DataOUT, BCNT = size of the received data OUT packet ($0 \leq$ BCNT $\leq 1\,024$), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
      PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
      These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.
5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 722 provides a flowchart of the above procedure.

Figure 722. Receive FIFO packet read



SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   – STUPCNT = 3 in OTG_DOEPTSIZx
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   – 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the

receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

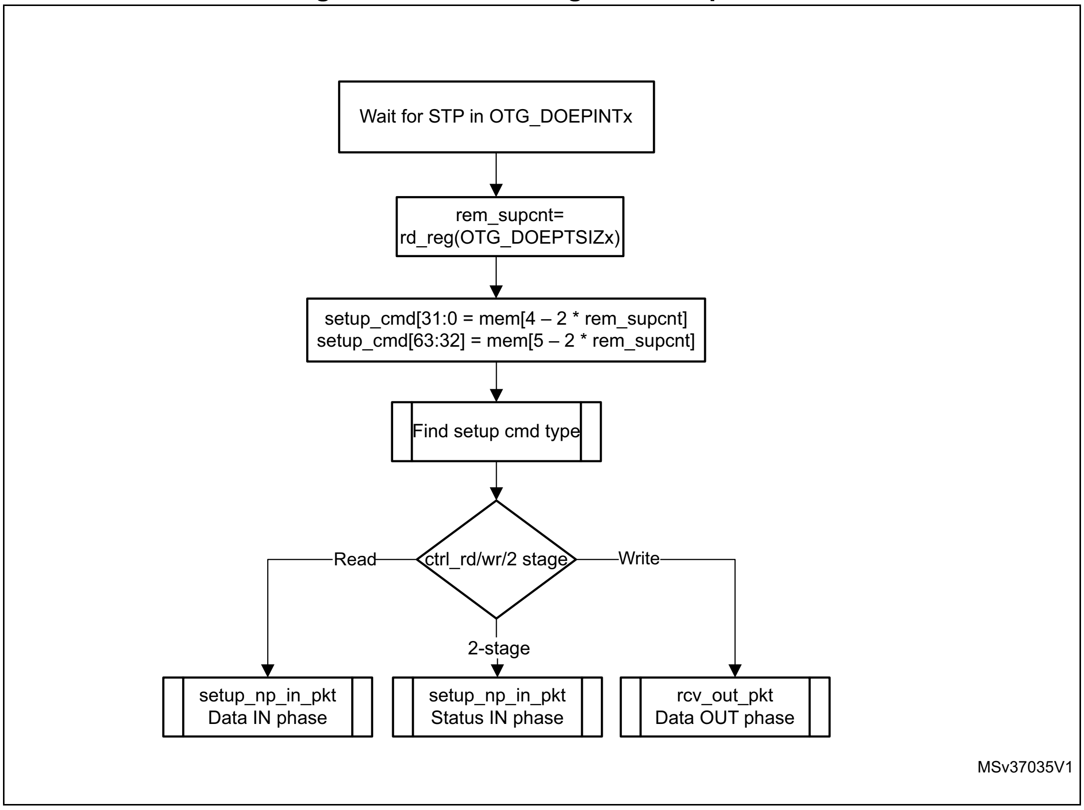
- **Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint’s NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first word contains control information used internally by the core
   – The second word contains the first 4 bytes of the SETUP command
   – The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   – On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 723. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_DOEPCTLx
   – SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_DOEPCTLx
   – EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_DCTL

- **Transfer Stop Programming for OUT endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Enable all OUT endpoints by setting
   – EPENA = 1 in all OTG_DOEPCTLx registers.
2. Flush the RxFIFO as follows
   – Poll OTG_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on OTG_GRSTCTL.RXFFLSH =1
   – Poll OTG_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “*Setting the global OUT NAK on page 3115*”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in OTG_DCTL
4. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers OTG_DOEPCTLx
   – SNAK = 1 in registers OTG_DOEPCTLx
6. Wait for the EPDIS interrupt in OTG_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in OTG_DOEPINTx indicates that the corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:
   – EPENA = 0 in registers OTG_DOEPCTLx
   – EPDIS = 0 in registers OTG_DOEPCTLx
   – SNAK = 0 in registers OTG_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint’s transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – transfer size[EPNUM] = n × (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = n
   – $n > 0$
3. On any OUT endpoint interrupt, the application must read the endpoint’s transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet ($0 \leq$ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   - EPENA = 1 in OTG_DOEPCTLx
   - CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   - This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   - EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2

     RXDPID = D2 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 3The number of USB packets in which this payload was received =
     Application programmed initial packet count – core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application may fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
   – When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   – When the isochronous OUT data packet is received with CRC errors
   – When the isochronous OUT token received by the core is corrupted
   – When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At

this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   – EPENA = 1 (in OTG_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).

     The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

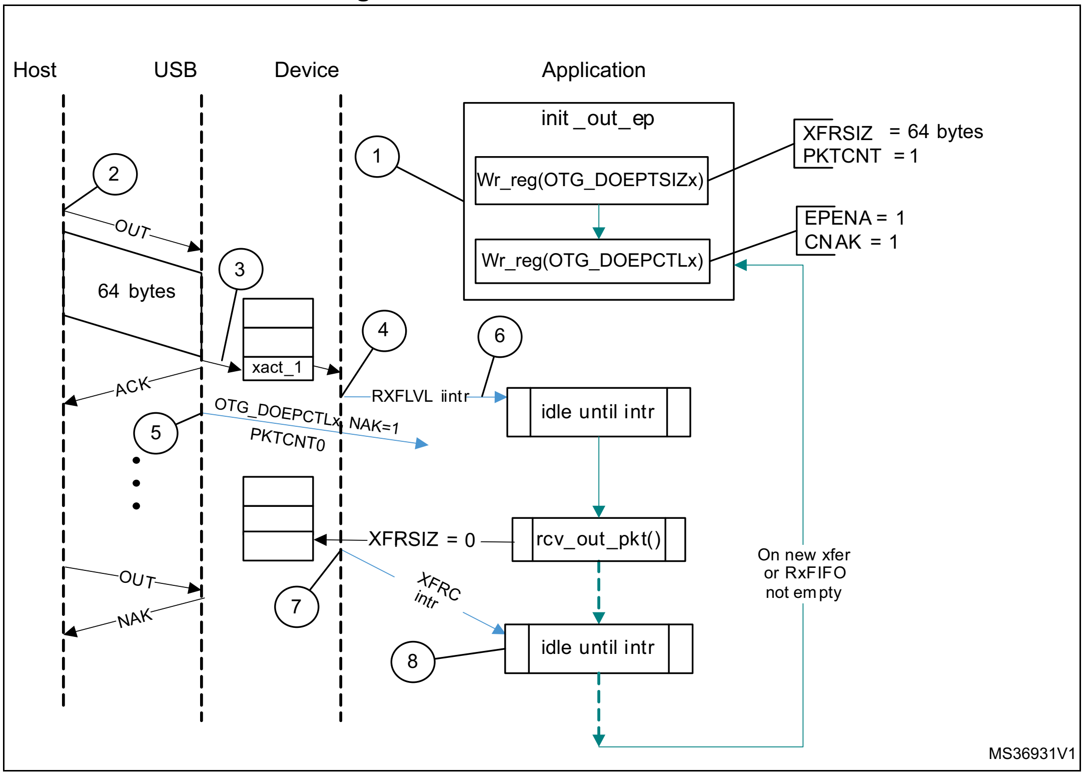
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 724 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 724. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   – In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   – To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).

   – CNAK = 1 in OTG_DIEPCTLx
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Transfer Stop Programming for IN endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Disable the IN endpoint by setting:
   – EPDIS = 1 in all OTG_DIEPCTLx registers
2. Wait for the EPDIS interrupt in OTG_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   – EPDIS = 0 in OTG_DIEPCTLx
   – EPENA = 0 in OTG_DIEPCTLx
3. Flush the TxFIFO by programming the following bits:
   – TXFFLSH = 1 in OTG_GRSTCTL
   – TXFNUM = “FIFO number specific to endpoint” in OTG_GRSTCTL
4. The application can start polling till TXFFLSH in OTG_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
     Transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = x
   – To transmit a single zero-length data packet:
     Transfer size[EPNUM] = 0
     Packet count[EPNUM] = 1
   – To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
     First transfer: transfer size[EPNUM] = x × MPSIZ[epnum]; packet count = n;
     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers on page 3125* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To

transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

transfer size[EPNUM] = $x \times$ MPSIZ[EPNUM] + sp
(where x is an integer ≥ 0, and $0 \leq sp <$ MPSIZ[EPNUM])

If (sp > 0), packet count[EPNUM] = $x + 1$
Otherwise, packet count[EPNUM] = x;

MCNT[EPNUM] = packet count[EPNUM]

- The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
- transfer size[EPNUM] = 0

  packet count[EPNUM] = 1

  MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT (in OTG_DIEPTSIZx)
   - If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   - Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   - A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   - A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO

mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.

   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints

5. The packet count for the endpoint is decremented by 1 under the following conditions:
   – For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   – For interrupt endpoints, when an ACK handshake is transmitted
   – When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it

eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   - SNAK = 1 in OTG_DIEPCTLx
   - EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt in OTG_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

58.15.7 Worst case response time

When the OTG controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token may come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

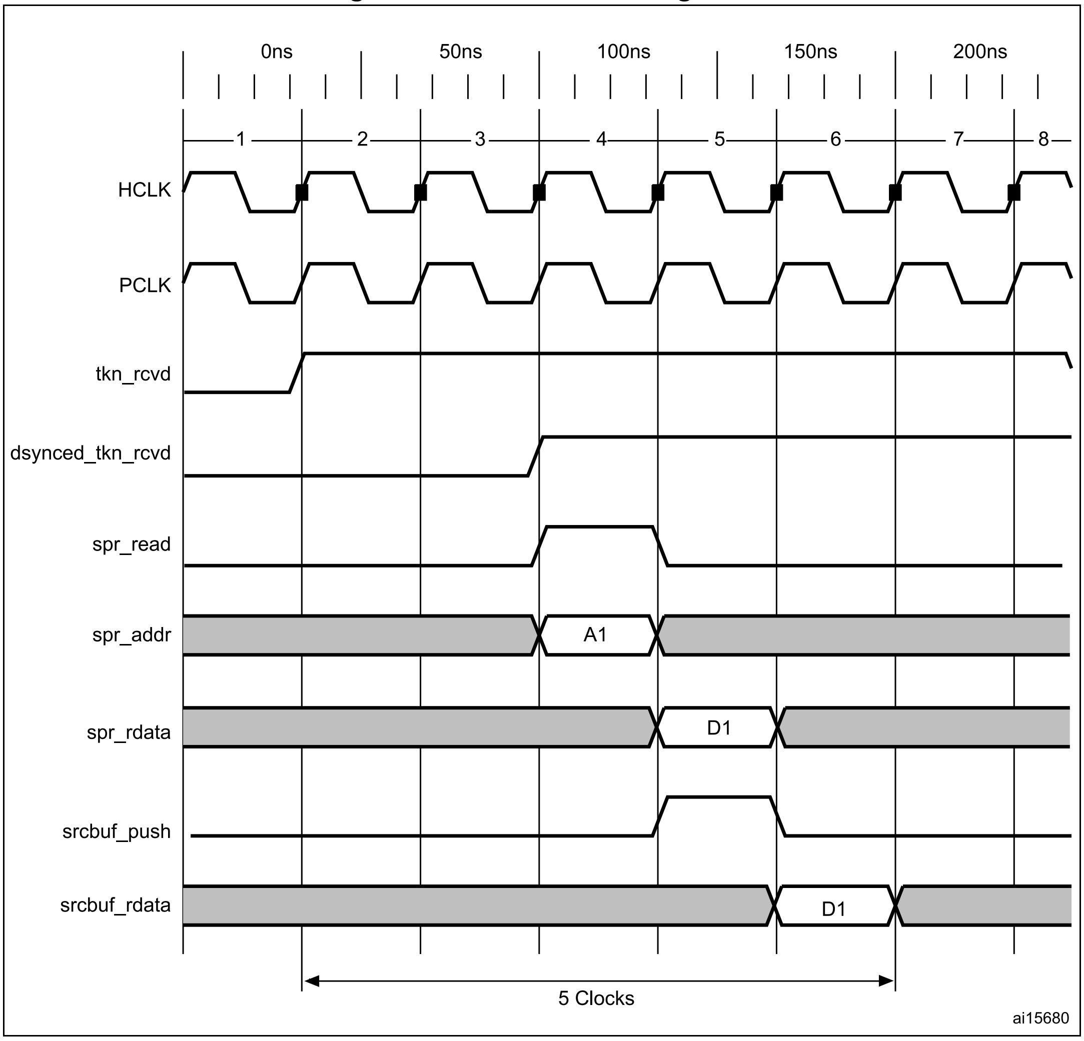
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 725 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 470: TRDT values*.

Figure 725. TRDT max timing case



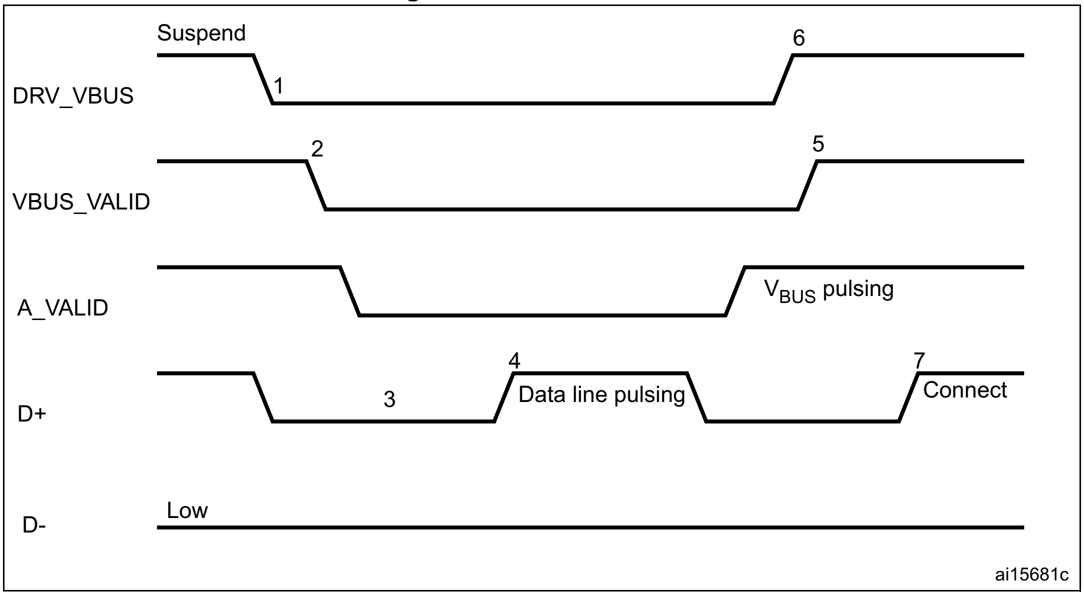
58.15.8 OTG programming model

The OTG controller is an OTG device supporting HNP and SRP. When the core is connected to an "A" plug, it is referred to as an A-device. When the core is connected to a "B" plug it is referred to as a B-device. In host mode, the OTG controller turns off V_{BUS} to conserve power. SRP is a method by which the B-device signals the A-device to turn on V_{BUS} power. A device must perform both data-line pulsing and V_{BUS} pulsing, but a host can detect either data-line pulsing or V_{BUS} pulsing for SRP. HNP is a method by which the B-device negotiates and switches to host role. In Negotiated mode after HNP, the B-device suspends the bus and reverts to the device role.

A-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG controller to detect SRP as an A-device.

Figure 726. A-device SRP



1. DRV_VBUS = V_{BUS} drive signal to the PHY
   VBUS_VALID = V_{BUS} valid signal from PHY
   A_VALID = A-peripheral V_{BUS} level signal to PHY
   D+ = Data plus line
   D- = Data minus line

The following points refer and describe the signal numeration shown in the *Figure 726*:

1. To save power, the application suspends and turns off port power when the bus is idle by writing the port suspend and port power bits in the host port control and status register.
2. PHY indicates port power off by deasserting the VBUS_VALID signal.
3. The device must detect SE0 for at least 2 ms to start SRP when V_{BUS} power is off.
4. To initiate SRP, the device turns on its data line pull-up resistor for 5 to 10 ms. The OTG controller detects data-line pulsing.
5. The device drives V_{BUS} above the A-device session valid (2.0 V minimum) for V_{BUS} pulsing.

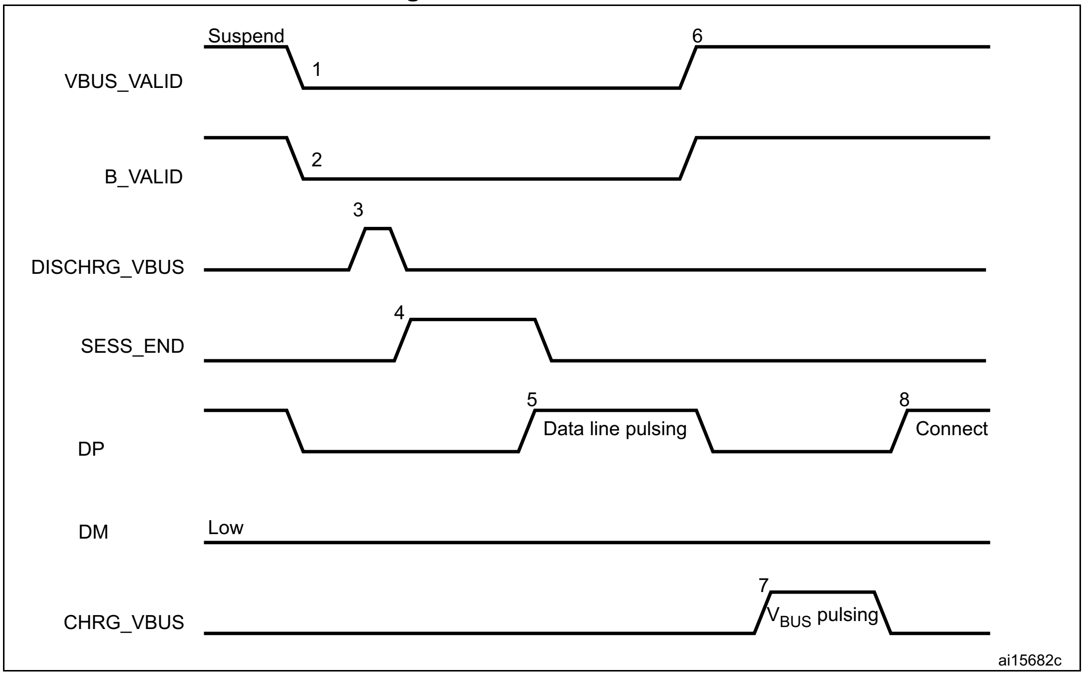
The OTG controller interrupts the application on detecting SRP. The session request detected bit is set in Global interrupt status register (SRQINT set in OTG_GINTSTS).

6. The application must service the session request detected interrupt and turn on the port power bit by writing the port power bit in the host port control and status register. The PHY indicates port power-on by asserting the VBUS_VALID signal.
7. When the USB is powered, the device connects, completing the SRP process.

B-device session request protocol

The application must set the SRP-capable bit in the core USB configuration register. This enables the OTG controller to initiate SRP as a B-device. SRP is a means by which the OTG controller can request a new session from the host.

Figure 727. B-device SRP



1. VBUS_VALID = V_{BUS} valid signal from PHY
B_VALID = B-peripheral valid session to PHY
DISCHRG_VBUS = discharge signal to PHY
SESS_END = session end signal to PHY
CHRG_VBUS = charge V_{BUS} signal to PHY
DP = Data plus line
DM = Data minus line

The following points refer and describe the signal numeration shown in the *Figure 727*:

1. To save power, the host suspends and turns off port power when the bus is idle.
The OTG controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG controller sets the USB suspend bit in the core interrupt register.
The OTG controller informs the PHY to discharge V_{BUS}.
2. The PHY indicates the session's end to the device. This is the initial condition for SRP. The OTG controller requires 2 ms of SE0 before initiating SRP.
For a USB 1.1 full-speed serial transceiver, the application must wait until V_{BUS} discharges to 0.2 V after BSVLD (in OTG_GOTGCTL) is deasserted. This discharge

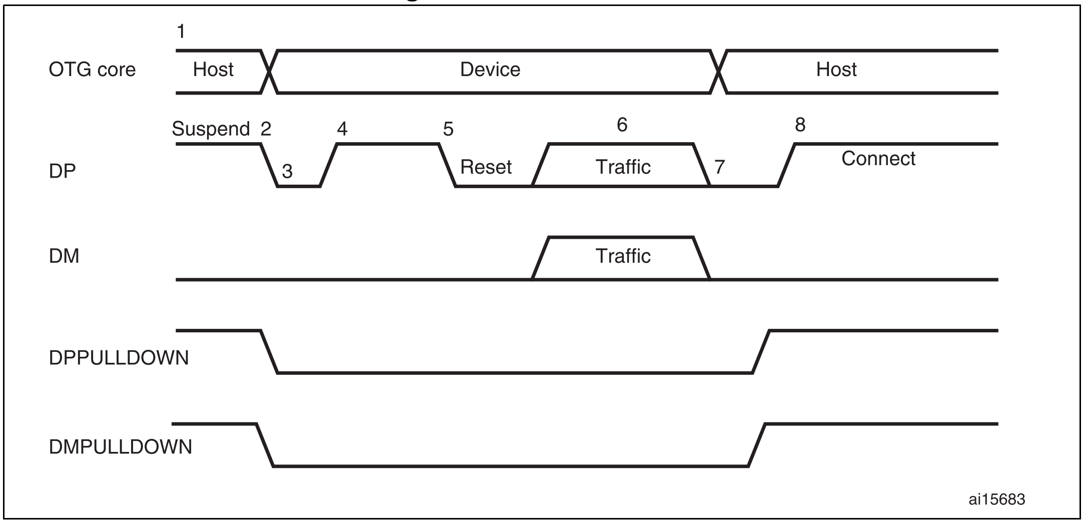
time can be obtained from the transceiver vendor and varies from one transceiver to another.

3. The OTG core informs the PHY to speed up V_{BUS} discharge.
4. The application initiates SRP by writing the session request bit in the OTG control and status register. The OTG controller perform data-line pulsing followed by V_{BUS} pulsing.
5. The host detects SRP from either the data-line or V_{BUS} pulsing, and turns on V_{BUS}. The PHY indicates V_{BUS} power-on to the device.
6. The OTG controller performs V_{BUS} pulsing.
   The host starts a new session by turning on V_{BUS}, indicating SRP success. The OTG controller interrupts the application by setting the session request success status change bit in the OTG interrupt status register. The application reads the session request success bit in the OTG control and status register.
7. When the USB is powered, the OTG controller connects, completing the SRP process.

A-device host negotiation protocol

HNP switches the USB host role from the A-device to the B-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG controller to perform HNP as an A-device.

Figure 728. A-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
   DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the *Figure 728*:

1. The OTG controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device’s ACK response indicates that the B-device

supports HNP. The application must set host Set HNP enable bit in the OTG control and status register to indicate to the OTG controller that the B-device supports HNP.

2. When it has finished using the bus, the application suspends by writing the port suspend bit in the host port control and status register.
3. When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.

   The OTG controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.

   The OTG controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_DP pull-up resistor to indicate a connect for B-device.

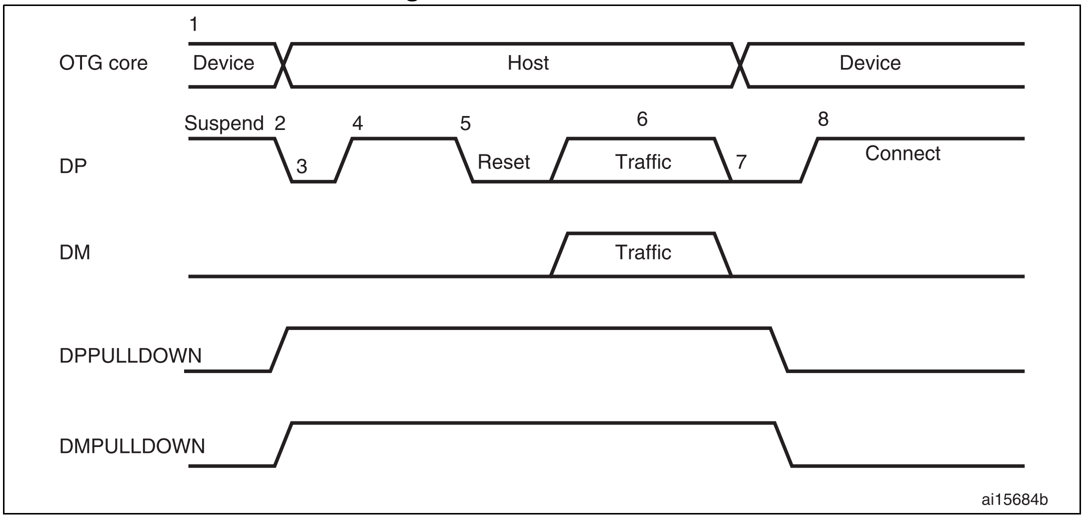
   The application must read the current mode bit in the OTG control and status register to determine device mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the OTG controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.

   The OTG controller sets the early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG controller sets the USB suspend bit in the core interrupt register.
6. In Negotiated mode, the OTG controller detects the suspend, disconnects, and switches back to the host role. The OTG controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG controller sets the connector ID status change interrupt in the OTG interrupt status register. The application must read the connector ID status in the OTG control and status register to determine the OTG controller operation as an A-device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

B-device host negotiation protocol

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the core USB configuration register to enable the OTG controller to perform HNP as a B-device.

Figure 729. B-device HNP



1. DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
   DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

The following points refer and describe the signal numeration shown in the *Figure 729*:

1. The A-device sends the SetFeature b_hnp_enable descriptor to enable HNP support. The OTG controller's ACK response indicates that it supports HNP. The application must set the device HNP enable bit in the OTG control and status register to indicate HNP support.

   The application sets the HNP request bit in the OTG control and status register to indicate to the OTG controller to initiate HNP.

2. When it has finished using the bus, the A-device suspends by writing the port suspend bit in the host port control and status register.

   The OTG controller sets the Early suspend bit in the core interrupt register after 3 ms of bus idleness. Following this, the OTG controller sets the USB suspend bit in the core interrupt register.

   The OTG controller disconnects and the A-device detects SE0 on the bus, indicating HNP. The OTG controller asserts the DP pull down and DM pull down in the PHY to indicate its assumption of the host role.

   The A-device responds by activating its OTG_DP pull-up resistor within 3 ms of detecting SE0. The OTG controller detects this as a connect.

   The OTG controller sets the host negotiation success status change interrupt in the OTG interrupt status register, indicating the HNP status. The application must read the host negotiation success bit in the OTG control and status register to determine host

negotiation success. The application must read the current Mode bit in the core interrupt register (OTG_GINTSTS) to determine host mode operation.
3. The application sets the reset bit (PRST in OTG_HPRT) and the OTG controller issues a USB reset and enumerates the A-device for data traffic.
4. The OTG controller continues the host role of initiating traffic, and when done, suspends the bus by writing the port suspend bit in the host port control and status register.
5. In Negotiated mode, when the A-device detects a suspend, it disconnects and switches back to the host role. The OTG controller deasserts the DP pull down and DM pull down in the PHY to indicate the assumption of the device role.
6. The application must read the current mode bit in the core interrupt (OTG_GINTSTS) register to determine the host mode operation.
7. The OTG controller connects, completing the HNP process.

59 USB HS PHY controller (USBPHYC)

59.1 USBPHYC introduction

The USBPHYC block is in the context of a system containing:

- Two port high-speed PHY
- USB high-speed OTG controller (single port)
- USB high-speed EHCI/OHCI host controller (two ports)

The USBPHYC enables the control and observation of a 2 port high-speed USB PHY configuration and status, and the sharing of one of those ports between different controllers.

59.2 USBPHYC main features

- Supports a two port high-speed PHY
- Enables one port to be switched between two possible controllers

59.3 USBPHYC functional description

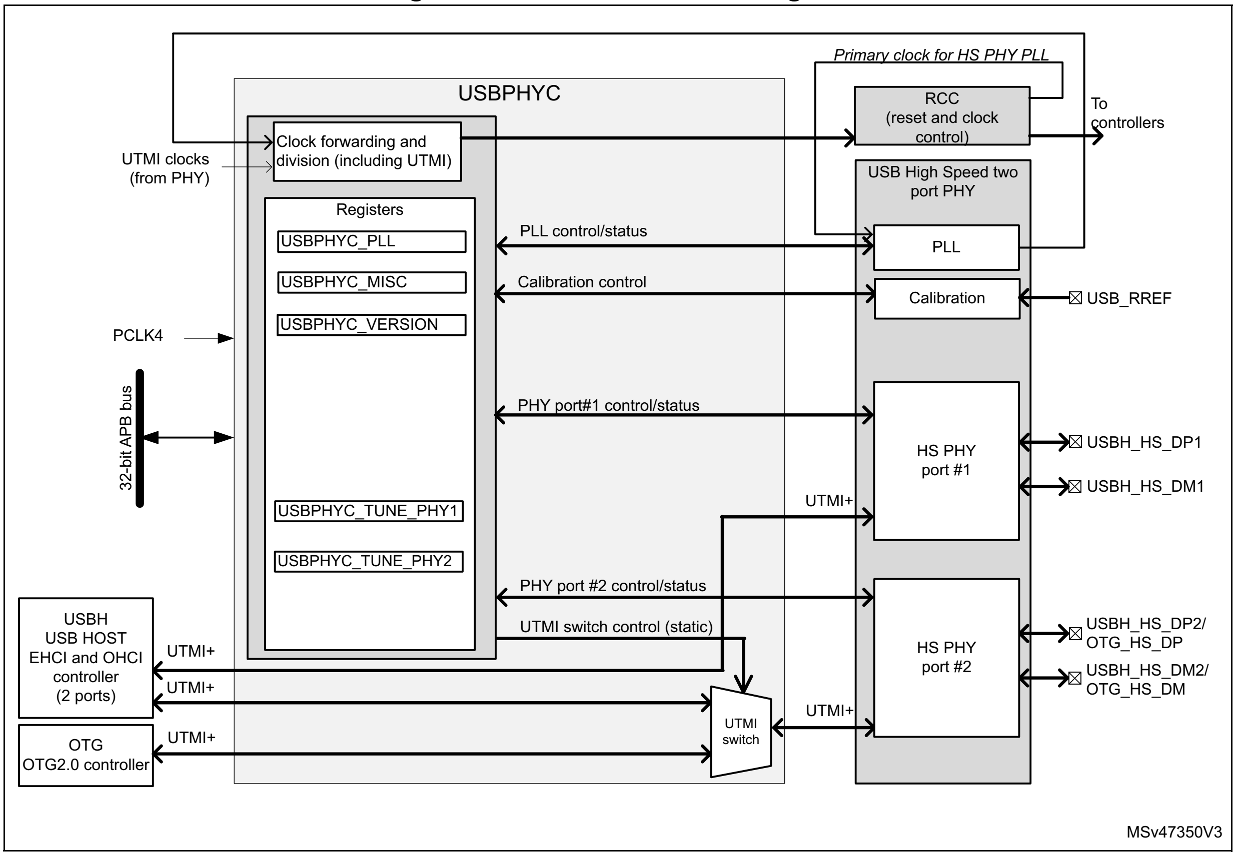
The USB HS PHY control block

- Controls the UTMI+ switch to share one port of the PHY between two controllers
- Sets the PLL values
- Sets other controls (and monitors) on the PHY

59.3.1 USBPHYC block diagram

The USBPHYC registers and main functions are shown in *Figure 730: USBPHYC block diagram*.

Figure 730. USBPHYC block diagram



59.3.2 USBPHYC reset and clocks

The USBPHYC is an APB peripheral. The USBPHYC logic is clocked directly by the PCLK4 clock of the APB bus interface. The registers are reset by the APB bus reset.

The list of other clocks available within the USBPHYC (not used for clocking the logic) is provided hereafter:

Input clocks (all coming from PLL section of the PHY):

- 60 MHz from PLL (for use as free running clock)
- 48 MHz from PLL (used for USB full speed PHY functions)
- UTMI clocks from each port (for controllers)

Output clocks used for muxing with other options and gating (all going to the RCC):

- 60 MHz
- 48 MHz
- 12 MHz (derived from 48 MHz)
- UTMI clocks (corresponding to each port), for distribution in the RCC to the relevant controllers (associated to each PHY port)

59.3.3 PLL control

The PLL should normally be configured according to requirements before setting PLLEN=1.

The fractional division mode is not enabled by default. In order to use it PLLFRACCTL should first be set and then the fractional part of the division control can be configured using different possible methods.

With the first method, PLLSTRB should be kept high for at least two cycles of the PLL input clock because the loading of control bits occurs at FBCLK transition (and these transitions cannot be monitored). In addition, the control bits should be stable for at least 3 clock cycles of the input clock after asserting PLLSTRB.

With the second method PLLSTRB should be bypassed by setting PLLSTRBBYP = 1. In this case, the control bits PLLFRACCTL and PLLFRACIN need to be static. The control bits should be changed only when the PLL is in power-down mode (PLLEN=0).

The loading of the control bits using PLLSTRB can be done only when the PLL is in coarse lock state (LOCKP = 1). When PLLSTRBBYP is low, the fractional bits are reset as soon as the PLL is unlocked. To load the fractional control bits, PLLSTRB should be asserted only after the lock is achieved (after a delay of 100 micro seconds).

The recommended approach for fractional division is thus to set all controls to the required values and PLLSTRBBYP=1, before setting PLLEN=1 to start the PLL.

59.4 USBPHYC registers

59.4.1 USBPHYC PLL control register (USBPHYC_PLL)

This register is used to control the PLL of the HS PHY.

Address offset: 0x000

Reset value: 0xC000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLDITHEN1	PLLDITHEN0	PLLFRACCTL	PLLSTRBYP	PLLSTRB	PLLEN	PLLFRACIN[15:6]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLFRACIN[5:0]						PLLODF[2:0]			PLLNDIV[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **PLLDITHEN1**: PLL dither 1 (rectangular)

This bit enables the rectangular dither for the PLL.
0: Enables the rectangular PDF dither input to SDM of PLL (not recommended)
1: Disables the rectangular PDF dither input to SDM of PLL

Bit 30 **PLLDITHEN0**: PLL dither 2 (triangular)

This bit enables the triangular dither for the PLL.
0: Enables the triangular PDF dither input to SDM of PLL (not recommended)
1: Disables the triangular PDF dither input to SDM of PLL

Bit 29 **PLLFRACCTL**: PLL fractional mode control

0: Fractional mode off
1: Fractional mode on

Bit 28 **PLLSTRBYP**: PLL strobe bypass

Bypass the strobe signal. If set to 1 before setting PLLEN=1, then the PLL will start up in fractional mode depending on the setting of PLLFRACCTL.
0: Do not bypass the strobe signal
1: Bypass the strobe signal

Bit 27 **PLLSTRB**: PLL strobe

This bit allows to control an asynchronous strobe signal provided as an input to the fractional controller. A rising edge of PLLSTRB signal indicates to the fractional controller that a new fractional word is to be loaded.
0: Strobe set to 0
1: Strobe set to 1

Bit 26 **PLLEN**: PLL enable

This bit enables the PLL.
0: PLL disabled
1: PLL enabled

Bits 25:10 **PLLFRACIN[15:0]**: PLL fractional input
PLLFRACIN is the fractional input of the PLL divider.

Bits 9:7 **PLLODF[2:0]**: PLL output division factor

Bits 6:0 **PLLNDIV[6:0]**: Loop division factor of PLL (integer part)

59.4.2 USBPHYC misc control register (USBPHYC_MISC)

This register is used to control the switch between controllers for the HS PHY.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPCKDIS[1:0]		SWITHOST
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:1 **PPCKDIS[1:0]**: Intelligent per HS PHY port clock gating control

This register is static.

0x0:No clock gating. PHY dedicated 60 MHz Port1/Port2 clocks are always delivered to target controller. PHY 48 MHz output clock is always delivered to target controller.

0x1:Intelligent clock gating for Port1. Port1 60 MHz input clock gated (Port1 is unused). Port2 dedicated 60 MHz is delivered in case of parallel data comm is requested on target controller (Host or OTG). PHY 48 MHz is delivered in case serial comm is requested on target controller (Host if SWITHOST = 1 else OTG).

0x2: Intelligent clock gating for Port2. Port2 60 MHz input clock gated (Port2 is unused). Port1 dedicated 60 MHz is delivered in case of parallel data comm is requested on target controller (Host). PHY 48 MHz is delivered in case serial comm is requested on target controller (Host if SWITHOST = 1 else OTG).

0x3:Intelligent clock gating for Port1 and Port2. Port1 dedicated 60 MHz is delivered in case of parallel data comm is requested on target (Host). Port2 dedicated 60 MHz is delivered in case of parallel data comm is requested on target (Host if SWITHOST = 1 else OTG). PHY 48 MHz output clock is delivered in case serial comm is requested on target controller (Host or OTG).

Bit 0 **SWITHOST**: Switch host

Controls main switch connecting USB PHY 2nd port between USB host controller and USB OTG controller. This register is static.

0: Select OTG controller for 2nd PHY port

1: Select Host controller for 2nd PHY port

59.4.3 USBPHYC PHY x MONITOR register (USBPHYC_MONITORx)

This register is used to control and monitor the debug interface of the HS PHY, port #x.

Address offset: 0x108 + 0x100 * (x-1), (x = 1 to 2)

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MON_SEL[4:0]					LOCKP	Res.	Res.	Res.
							rw	rw	rw	rw	rw	r			

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:4 **MON_SEL[4:0]**: Choice of debug output
31: Enable lock bit visibility
Others: Reserved

Bit 3 **LOCKP**: PLL lock visibility (when MON_SEL is set to 31)

Bits 2:0 Reserved, must be kept at reset value.

59.4.4 USBPHYC PHY x TUNE register (USBPHYC_TUNEx)

This register is used to control the tune interface of the HS PHY, port #x.

Address offset: 0x10C + 0x100 * (x-1), (x = 1 to 2)

Reset value: 0x0407 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STAGSEL	SHTCCTCTLPROT	HSFALLPREEM	HSRXOFF[1:0]		HDRXGNEQEN	SQLCHCTL[1:0]		OTPCOMP[4:1]			
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTP COMP [0]	HSDRVCHKZTRM[1:0]		HSDRVCHKITRM[3:0]				HSDRVRFRED	FSDRVRFADJ	HSDRVCURINCR	HSDRVDCLEV	HSDRVDCCUR	HSDRVSLEW	LFSCAPEN	INCURRINT	INCURREN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STAGSEL**: Staggering selection

This bit enables the HS Tx staggering.

0: Disable the basic staggering in HS Tx mode

1: Enable the basic staggering in HS Tx mode

Bit 26 **SHTCCTCTLPROT**: Short circuit control protection

This bit enables the short circuit protection circuitry in LS/FS driver.

0: Short circuit protection disabled

1: Short circuit protection enabled

Bit 25 **HSFALLPREEM**: HS fall time pre-emphasis

This bit enables the HS fall time control of single ended signals during pre-emphasis:

0: Control On

1: Control Off

Bits 24:23 **HSRXOFF[1:0]**: HS receiver offset adjustment

0x0: No offset

0x1: Offset +5 mV

0x2: Offset +10 mV

0x3: Offset -5 mV

Bit 22 **HDRXGNEQEN**: Enable HS Rx gain equalizer

0: Disable the gain equalizer

1: Enable the gain equalizer

Bits 21:20 **SQLCHCTL[1:0]**: Squelch control
This bitfield adjusts the squelch DC threshold value.
0x0: No shift in threshold
0x1: Squelch DC threshold shift by +7 mV
0x2: Squelch DC threshold shift by -5 mV
0x3: Squelch DC threshold shift by +14 mV

Bits 19:15 **OTPCOMP[4:0]**: OTP compensation code

Bits 14:13 **HSDRVCHKZTRM[1:0]**: HS driver choke impedance trim
This bitfield controls the PHY bus HS driver impedance tuning for choke compensation.
0x0: No impedance offset
0x1: Reduce the impedance by 2 Ω
0x2: Reduce the impedance by 4 Ω
0x3: Reduce the impedance by 6 Ω

Bits 12:9 **HSDRVCHKITRM[3:0]**: HS driver choke current trim
This bitfield controls the HS driver current trimming pins for choke compensation.
0x0: 18.87 mA target current / nominal + 0%
0x1: 19.165 mA target current / nominal + 1.56%
0x2: 19.46 mA target current / nominal + 3.12%
0x3: 19.755 mA target current / nominal + 4.68%
0x4: 20.05 mA target current / nominal + 6.24%
0x5: 20.345 mA target current / nominal + 7.8%
0x6: 20.64 mA target current / nominal + 9.36%
0x7: 20.935 mA target current / nominal + 10.92%
0x8: 21.23 mA target current / nominal + 12.48%
0x9: 21.525 mA target current / nominal + 14.04%
0xA: 21.82 mA target current / nominal + 15.6%
0xB: 22.115 mA target current / nominal + 17.16%
0xC: 22.458 mA target current / nominal + 19.01%
0xD: 22.755 mA target current / nominal + 20.58%
0xE: 23.052 mA target current / nominal + 22.16%
0xF: 23.348 mA target current / nominal + 23.73%

Bit 8 **HSDRVRFRED**: High-speed rise-fall reduction enable
0: Default rise/fall time
1: Increases the rise/fall time by 20%

Bit 7 **FSDRVRFADJ**: Tuning pin to adjust the full speed rise/fall time
0: Disables the full speed rise/fall tuning option
1: Enables the full speed rise/fall tuning option

Bit 6 **HSDRVCURINCR**: Enable the HS driver current increase feature
0: Disables the HSDRVDCLEV feature
1: Enables the HSDRVDCLEV feature

Bit 5 **HSDRVDCLEV**: HS driver DC level
This bit allows to increase the HS driver DC level. I is not applicable during the HS test J and test K data transfer.
0: Increases the HS driver DC level by 5 to 7 mV if HSDRVCURINCR = '1'
1: Increases the HS driver DC level by 10 to 14 mV if HSDRVCURINCR = '1'

Bit 4 **HSDRVDCCUR**: HS driver DC level
This bit allows to decrease the HS driver DC level.
0: Keeps the normal HS driver DC level
1: Decreases the HS driver DC level by 5 to 7 mV

Bit 3 **HSDRVSLEW**: HS driver slew rate
This bit controls the HS driver slew rate.
0: Keeps the normal slew rate
1: Slows the driver slew rate by 10%

Bit 2 **LFSCAPEN**: Low full speed enable
This bit enables the low full speed feedback capacitor.
0: Disables the feedback capacitor.
1: Enables the feedback capacitor.

Bit 1 **INCURRINT**: Current boosting value
This bit controls the PHY current boosting value.
0: Provides a current boosting of 1mA if INCURREN = '1'
1: Provides a current boosting of 2mA if INCURREN = '1'

Bit 0 **INCURREN**:
The bit enables the current boosting function.
0: Disables the current boosting
1: Enables the current boosting

59.4.5 USBPHYC VERSION register (USBPHYC_VERR)

This register defines the version of this IP.

Address offset: 0xFFC

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision

Bits 3:0 **MINREV[3:0]**: Minor revision

59.4.6 USBPHYC register map

Table 473. USBPHYC register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	USBPHYC_PLL	PLLDITHEN1	PLLDITHEN0	PLLFRACCT	PLLSTRBYP	PLLSTRB	PLLEN	PLLFRACIN[15:0]																PLLODF[2:0]			PLLNDIV[6:0]						
	Reset value	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x008	USBPHYC_MISC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPCKDIS[1:0]		SWITHOST
	Reset value																														0	0	0
0x00C-0x0104	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x108	USBPHYC_ MONITOR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MON_SEL [4:0]					LOCKP	Res.	Res.	Res.
	Reset value																								0	0	0	0	0	X			
0x10C	USBPHYC_TUNE1	Res.	Res.	Res.	Res.	STAGSEL	SHTCCTCTLPROT	HSFALLPREEM	HSRXOFF[1:0]		HDRXGNEQEN	SQLCHCTL[1:0]		OTPCOMP[4:0]					HSDRVCHKZTRM[1:0]		HSDRVCHKITRM[3:0]				HSDRVRFRED	FSDRVRFADJ	HSDRVCURINCR	HSDRVDCLEV	HSDRVDCCUR	HSDRVSLEW	LFSCAPEN	INCURRINT	INCURREN
	Reset value					0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x110 - 0x204	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x208	USBPHYC_ MONITOR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MON_SEL [4:0]					LOCKP	Res.	Res.	Res.
	Reset value																								0	0	0	0	0	X			
0x20C	USBPHYC_TUNE2	Res.	Res.	Res.	Res.	STAGSEL	SHTCCTCTLPROT	HSFALLPREEM	HSRXOFF[1:0]		HDRXGNEQEN	SQLCHCTL[1:0]		OTPCOMP[4:0]					HSDRVCHKZTRM[1:0]		HSDRVCHKITRM[3:0]				HSDRVRFRED	FSDRVRFADJ	HSDRVCURINCR	HSDRVDCLEV	HSDRVDCCUR	HSDRVSLEW	LFSCAPEN	INCURRINT	INCURREN
	Reset value					0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x210 - 0xFF8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 473. USBPHYC register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	USBPHYC_VERR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
	Reset value																									0	0	0	1	0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

60 USB_EHCI/USB_OHCI high/full-speed link controller (USBH)

60.1 Introduction

The USBH is a USB host controller supporting the standard registers used for full-speed (OHCI controller) and high-speed (EHCI controller).

60.2 USBH main features

- Two physical ports supported using an on-chip 2-port high-speed PHY
- Supports OHCI model for full- and low-speed operation
- Supports EHCI model for high-speed operation
- LPM (link power management) support
- DMA engine with local 512 Byte packet buffer

The standards supported are:

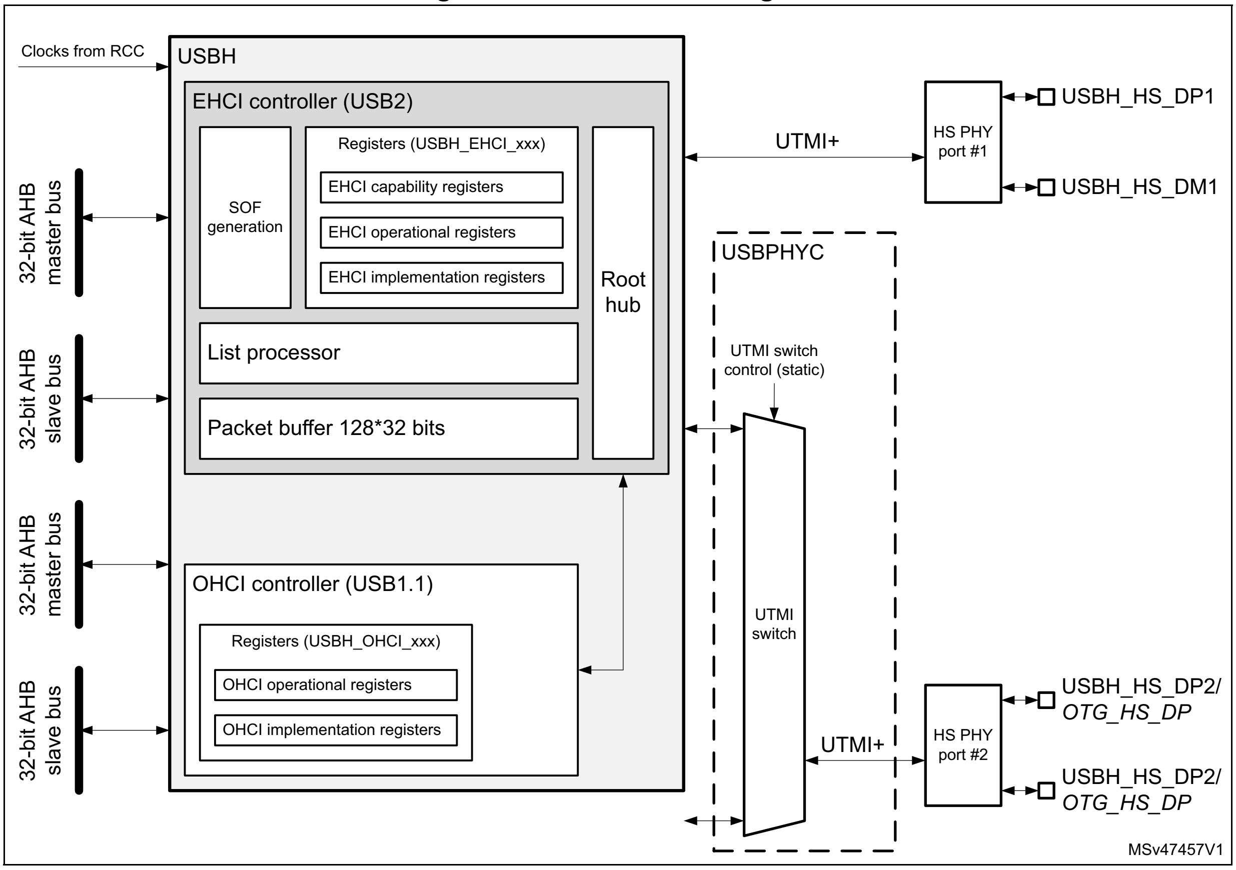
- Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 1.0, Intel Corp., March 12, 2002
  - http://www.intel.com
- EHCI 1.1 Addendum, Version 1.1, Intel Corp., August 2008
  - http://www.intel.com
- Open Host Controller Interface Specification for USB, Release 1.0a, Compaq, Inc., Microsoft Corp., National Semiconductor Corp., September 14, 1999
  - http://www.usb.org
- Universal Serial Bus Specification, Revision 2.0, USB Implementers Forum, Inc., April 27, 2000
  - http://www.usb.org
- USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, Revision 1.05, Intel Corp., March 29, 2001
  - http://www.intel.com
- UTMI+ Specification, Revision 1.0, Philips Semiconductor, February 25, 2004
  - http://www.nxp.com

60.3 USBH functional description

60.3.1 USBH block diagram

The USBH block diagram is shown below:

Figure 731. USBH block diagram



60.3.2 USBH reset and clocks

USB requires both a functional AHB bus clock and functional USB clocks. The USB clocks are generated in USBPHYC (programming and enabling HS PHY's dedicated PLL). The RCC (reset and clock controller) controls clock gating and some multiplexing options. Refer to *Section 59: USB HS PHY controller (USBPHYC)* and *Section 10: Reset and clock control (RCC)* for full details on clock and reset control.

Note: In order to access USBH_OHCI registers it is necessary to activate the USB clocks by enabling the PLL controlled by USBPHYC.

60.3.3 Description of USBH packet buffer

The packet buffer provides storage and control for IN/OUT data. During OUT transactions, the system memory controller (inside the root hub) fetches data from the system memory and writes it here. During IN transactions, the root hub block writes the data. The packet buffer has a maximum depth of 512 bytes.

Packet buffer depth selection influences the system latency and bandwidth allocated to the EHCI host controller. For example, if the packet buffer depth is set to the maximum possible value i.e. 512 bytes, then for an IN transfer with USB packet size of 1024 bytes, there will be two (1024/512) data transfers on the AHB bus.

60.3.4 Description of USBH root hub

The USBH root hub propagates Reset and Resume signals to downstream ports and handles port connections and disconnections. In addition, the root hub is implemented with port router functionality to route the ports to either the EHCI host controller or OHCI host controller. The root hub operates on the local PHY clock, which operates on a free-running 60 MHz clock, and the clock source from each physical port, which operate at 60 MHz with an 8-bit interface.

60.4 USBH interrupts

Independent interrupts are generated from each controller:

- OHCI: usbh_ohci_it
  - HCI bus general interrupt (covers all 9 interrupt conditions covered by standard OHCI register HcInterruptStatus)
- EHCI: usbh_ehci_it
  - Interrupt async address
  - Host system error
  - Frame list rollover
  - Port change
  - USB error
  - USB interrupt

60.5 USBH registers

Note that USBH has two physical AHB slaves:

- USBH_OHCI (mapped to USBH_OHCI_BASE)
- USBH_EHCI (mapped to USBH_OHCI_BASE + 0x1000 offset)

USBH described here covers both slaves and to simplify, all offsets are defined here relative to the lowest system address base (i.e. that of the USBH_OHCI slave).

60.5.1 USBH OHCI standard registers

The OHCI core is implemented according USB 1.1 specification, release 1.0a. The standard OHCI operational registers are listed hereafter and in *Table 474: USBH register map and reset values* however their detailed description has been omitted in this document. Please refer to the standard documentation for the register description. In the following list of registers, the prefix represents the address offset with respect to USBH_OHCI base address.

- [0x0000]: USBH_OHCI_HCREVISION
- [0x0004]: USBH_OHCI_HCCONTROL
- [0x0008]: USBH_OHCI_HCCOMMANDSTATUS
- [0x000C]: USBH_OHCI_HCINTERRUPTSTATUS
- [0x0010]: USBH_OHCI_HCINTERRUPTENABLE
- [0x0014]: USBH_OHCI_HCINTERRUPTDISABLE
- [0x0018]: USBH_OHCI_HCHCCA
- [0x001C]: USBH_OHCI_HCPERIODCURRENTED
- [0x0020]: USBH_OHCI_HCCONTROLHEADED
- [0x0024]: USBH_OHCI_HCCONTROLCURRENTED
- [0x0028]: USBH_OHCI_HCBULKHEADED
- [0x002C]: USBH_OHCI_HCBULKCURRENTED
- [0x0030]: USBH_OHCI_HCDONEHEAD
- [0x0034]: USBH_OHCI_HCFMINTERVAL
- [0x0038]: USBH_OHCI_HCFMREMAINING
- [0x003C]: USBH_OHCI_HCFMNUMBER
- [0x0040]: USBH_OHCI_PERIODICSTART
- [0x0044]: USBH_OHCI_HCLSTHRESHOLD
- [0x0048]: USBH_OHCI_HCRHDESCRIPTORA
- [0x004C]: USBH_OHCI_HCRHDESCRIPTORB
- [0x0050]: USBH_OHCI_HCRHSTATUS
- [0x0054]: USBH_OHCI_HCRHPORTSTATUS1
- [0x0058]: USBH_OHCI_HCRHPORTSTATUS2

60.5.2 USBH OHCI implementation register #6 (USBH_OHCI_INSNREG06)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HBURST[2:0]			EXP_BEATS[4:0]					SUC_BEATS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **AHB_ECAP**: AHB error captured
Indicator that an AHB error was encountered and values were captured. To clear this field the application must write a 0 to this register.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:9 **HBURST[2:0]**: HBURST value of the control phase at which the AHB error occurred

Bits 8:4 **EXP_BEATS[4:0]:** Number of expected beats
Number of beats expected in the burst where the AHB error occurred. Valid values are 0 and 4.
0: When no error, 0 is written to **EXP_BEATS[4:0]**.
4: When INCR4 and an error has occurred, 4 is written to **EXP_BEATS[4:0]**.
Other values except 4 are not written to **EXP_BEATS[4:0]**.

Bits 3:0 **SUC_BEATS[3:0]**: Number of successful beats
Number of successfully completed beats in the current burst before the AHB error has occurred.

60.5.3 USBH OHCI implementation register #7 (USBH_OHCI_INSNREG07)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_MST_ERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB_MST_ERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AHB_MST_ERR[31:0]**: AHB master error address
AHB address of the control phase at which the AHB error has occurred.

60.5.4 USBH EHCI standard registers

The EHCI core is implemented according USB 1.1 specification release 1.0a, and EHCI Addendum 1.1. The standard EHCI registers are listed hereafter and in *Table 474: USBH register map and reset values* however their detailed description has been omitted in this document. Please refer to the standard documentation for the register description. In the following list of registers, the prefix represents the address offset with respect to USBH_OHCI base address.

- EHCI capability registers:
  - [0x1000]: USBH_EHCI_HCICAPLENGTH
  - [0x1004]: USBH_EHCI_HCSPARAMS
  - [0x1008]: USBH_EHCI_HCCPARAMS
- EHCI functional registers
  - [0x1010]: USBH_EHCI_USBCMD
  - [0x1014]: USBH_EHCI_USBSTS
  - [0x1018]: USBH_EHCI_USBINTR
  - [0x101C]: USBH_EHCI_FRINDEX
  - [0x1024]: USBH_EHCI_PERIODICLISTBASE
  - [0x1028]: USBH_EHCI_ASYNCLISTADDR
  - [0x1050]: USBH_EHCI_CONFIGFLAG
  - [0x1054]: USBH_EHCI_PORTSC1
  - [0x1058]: USBH_EHCI_PORTSC2

60.5.5 USBH EHCI implementation register #1 (USBH_EHCI_INSNREG01)

This register controls the programmable packet buffer OUT/IN thresholds. The value specified here is the number of 32-bit words.

The OUT threshold is used to start the USB transfer as soon as the OUT threshold amount of data is fetched from system memory. It is also used to disconnect the data fetch, if the required space to hold the threshold amount of data is not available in the packet buffer.

The IN threshold is used to start the memory transfer as soon as the IN threshold amount of data is available in the packet buffer. It is also used to disconnect the data write, if the threshold amount of data is not available in the packet buffer.

The minimum OUT and IN threshold amount that can be programmed through INSN registers is 16 bytes. The minimum threshold amount that can be programmed is the highest possible INCRX burst value i.e. INCR16. The minimum OUT and IN threshold value should be 64 bytes (16×32-bit words).

OUT and IN threshold values can be equal to the packet buffer depth only when isochronous/interrupt transactions are not initiated by the host controller.

Note: The break memory transfer bit is always enabled.

Note: Depending on packet buffer depth settings, the most significant bits may be unused.

Note: Power on reset sets these registers to reset values, soft reset does not affect these registers.

Address offset: 0x1094

Reset value: 0x0020 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT_THRESHOLD[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN_THRESHOLD[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **OUT_THRESHOLD[7:0]**: The value specified here is the number of 32-bit words

The OUT threshold is used to start the USB transfer as soon as the OUT_THRESHOLD amount of data is fetched from system memory. It is also used to disconnect the data fetch, if the required space to hold the threshold amount of data is not available in the packet buffer.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IN_THRESHOLD[7:0]**: The value specified here is the number of 32-bit words

The IN threshold is used to start the memory transfer as soon as the IN_THRESHOLD amount of data is available in the packet buffer. It is also used to disconnect the data write, if the threshold amount of data is not available in the packet buffer.

60.5.6 USBH EHCI implementation register #2 (USBH_EHCI_INSNREG02)

Programmable packet buffer depth.

Address offset: 0x1098

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT_BUF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PKT_BUF[7:0]**: Programmable packet buffer depth

Programmable packet buffer depth.
The value specified here represents the number of 32-bit words.
Maximum valid packet buffer depth is 128 (32-bit words).

60.5.7 USBH EHCI implementation register #3 (USBH_EHCI_INSNREG03)

Address offset: 0x109C

Reset value: 0x0001 2001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE_LS_GLITCH
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	EN_CLK256_CHECK	TESTSE_NAK	TX_TRA_DELAY[2:0]			FRM_LST_FETCH	TIME_AVAIL_OFF[7:0]								BRK_MEM_TRANS
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **ENABLE_LS_GLITCH**: Enable/disable enhancement for line state glitch

By default this enhancement is enabled.

0: disabled

1: enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **EN_CLK256_CHECK**: Enable 256 clock checking

This bit controls the end of resume sequence of the EHCI host controller. By default, the value of this bit is 0 and during the end of resume sequence, the host controller waits for SE0 on the linestate before switching the PHY to high-speed.

When set to 1, during the end of resume sequence, the controller waits for SE0 or 256 clocks before switching the PHY to high-speed.

Setting this bit to 1, enables the 256 clock checking. Some of the UTMI PHYs do not present SE0 on the linestate during the end of resume sequence. For such PHYs, this bit should be set, so that the core does not wait forever for SE0.

This bit should be set only during initialization.

Bit 13 **TESTSE_NAK**: TestSE NAK

Ignore linestate during TestSE0 Nak

When set to 1 (default), the core ignores the linestate checking when transmitting SOF during the SE0_NAK test mode.

When set to 0, the port state machine disables the port if it does not find the linestate to be in SE0 when transmitting SOF during the SE0_NAK testing. While doing impedance measurement during the SE0_NAK testing, the linestate could go to non SE0 forcing the core to disable the port. This bit is used to control the port behavior during this state.

Bits 12:10 **TX_TRA_DELAY[2:0]**: Tx-Tx turnaround delay add on

This field specifies the extra delays in phy_clks to be added to the "Transmit to Transmit turnaround delay" value maintained in the core. The default value of this register field is 0. This default value of 0 is sufficient for most PHYs. But for some PHYs that put wait states during the token packet, it may be required to program a value greater than 0 to meet the transmit to transmit minimum turnaround time. The recommendation is to use the default value of 0 and change it only if there is an issue with minimum transmit-to-transmit turnaround time. This value should be programmed during core initialization and should not be changed afterwards.

Bit 9 **FRM_LST_FETCH**:

Setting this bit will force the host controller to fetch the periodic frame list in every microframe of a frame. If not set, then the periodic frame list will be fetched only in microframe 0 of every frame.

The default is 0 (not set). This bit can be changed only during core initialization and should not be changed afterwards.

Bits 8:1 **TIME_AVAIL_OFF[7:0]**: Time available offset

This value indicates the additional number of bytes to be accommodated for the time-available calculation.

The USB traffic on the bus can be started only when sufficient time is available to complete the packet within the EOF1 point. Refer to the USB 2.0 specification for details of the EOF1 point.

This time-available calculation is done in the hardware, and can be further offset by programming a value in this location.

Note: Time-available calculation is added for future flexibility. The application is not required to program this field by default.

Bit 0 **BRK_MEM_TRANS**: Break memory transfer

Used in conjunction with EHCI_H_INSNREG01 to enable breaking memory transactions into chunks once the OUT/IN threshold value is reached. Value is fixed to 1.

60.5.8 USBH EHCI implementation register #4 (USBH_EHCI_INSNREG04)

Address offset: 0x10A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPEND_SIGNAL	Res.	Res.	Res.	HCCPARAMS_WRT	HCSPARAMS_WRT
										rw				rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **SUSPEND_SIGNAL**: Suspend signal

0: By default, the automatic feature is enabled. The suspend signal is de-asserted (logic level 1) when run/stop is reset by software, but the HCH (hchalted) bit is not yet set.

1: Disables the automatic feature, which takes all ports out of suspend when software clears the run/stop bit. This is for backward compatibility.

For systems where the host is halted without waking up all ports out of suspend, the port can become stuck because the clock coming from the PHY is not running when the halt is programmed. To avoid this, the EHCI automatically pulls ports out of suspend when the host is halted by software. This bit is used to disable this automatic function.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **HCCPARAMS_WRT**:

When this bit is 1, the USBH_EHCI_HCCPARAMS register bits 17, 15:4, and 2:0 become writable.

Bit 0 **HCSPARAMS_WRT**:

When this bit is 1, the USBH_EHCI_HCSPARAMS register becomes writable.

60.5.9 USBH EHCI implementation register #5 (USBH_EHCI_INSNREG05)

Address offset: 0x10A4

Reset value: 0x0000 1000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSY	VPORT [3]
														r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VPORT[2:0]			VCONTROL_LDM	VCONTROL[3:0]				VSTATUS[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **VBUSY**: UTMI VBUSY

VBusy (read only). Hardware indicator that a write to this register has occurred and the hardware is currently processing the operation defined by the data written. When processing is finished, this bit is cleared.

Bits 16:13 **VPORT[3:0]**: UTMI VPORT

VPort (read write). Valid values for write are only 1 and 2. No other values should be written to this field. Supposing an illegal value, say 4, is written, any subsequent writes to this register are ignored and the read value will always be the initial illegal value written (in this case 4).

Bit 12 **VCONTROL_LDM:** UTMI VCONTROLLoad (vendor control load)
Refer to the UTMI specification for detailed usage.
0: Load
1: NOP

Bits 11:8 **VCONTROL[3:0]**: UTMI VCONTROL (vendor control)
Vendor defined 4-bit parallel input bus. Refer to the UTMI specification for detailed usage.

Bits 7:0 **VSTATUS[7:0]**: UTMI VStatus (vendor status)
Refer to the UTMI specification for detailed usage.

60.5.10 USBH EHCI implementation register #6 (USBH_EHCI_INSNREG06)

Address offset: 0x10A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HBURST[2:0]			EXP_BEATS[4:0]					SUC_BEATS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **AHB_ECAP**: AHB error captured
Indicator that an AHB error was encountered and values were captured. To clear this field the application must write a 0 to this bit.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:9 **HBURST[2:0]**: HBURST value of the control phase at which the AHB error occurred

Bits 8:4 **EXP_BEATS[4:0]:** Number of expected beats
Number of beats expected in the burst at which the AHB error has occurred. Valid values are 0 to 16.
0: When no error, 0 is written to **EXP_BEATS[4:0]**.
4: When INCR4 and error has occurred, 4 is written to **EXP_BEATS[4:0]**
8: When INCR8 and error has occurred, 8 is written to **EXP_BEATS[4:0]**
16: When INCR16 and error has occurred,16 is written to **EXP_BEATS[4:0]**
Other values except 4, 8, and 16 are not written to **EXP_BEATS[4:0]**

Bits 3:0 **SUC_BEATS[3:0]**: Number of successful beats
Number of successfully-completed beats in the current burst before the AHB error has occurred.

60.5.11 USBH EHCI implementation register #7 (USBH_EHCI_INSNREG07)

Address offset: 0x10AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB_MST_ERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AHB_MST_ERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AHB_MST_ERR[31:0]**: AHB master error address
AHB address of the control phase at which the AHB error has occurred.

60.5.12 USBH register map

Table 474. USBH register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**USBH_OHCI_HCREVISION**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV[7:0]							
	Reset value																									0	0	0	1	0	0	0	0
0x004	**USBH_OHCI_HCCONTROL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWE	RWC	IR	HCFS[1:0]		BLE	CLE	IE	PLE	CBSR[1:0]	
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x008	**USBH_OHCI_HCCOMMANDSTATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR	BLF	CLF	HCR
	Reset value															0	0													0	0	0	0
0x00C	**USBH_OHCI_HC INTERRUPT STATUS**	Res.	OC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RHSC	FNO	UE	RD	SF	WDH	SO
	Reset value		0																								0	0	0	0	0	0	0
0x010	**USBH_OHCI_HC INTERRUPT ENABLE**	MIE	OC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RHSC	FNO	UE	RD	SF	WDH	SO
	Reset value	0	0																								0	0	0	0	0	0	0
0x014	**USBH_OHCI_HC INTERRUPT DISABLE**	MIE	OC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RHSC	FNO	UE	RD	SF	WDH	SO
	Reset value	0	0																								0	0	0	0	0	0	0

Table 474. USBH register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	USBH_OHCI_ HCHCCA	HCCA[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x01C	USBH_OHCI_ HCPERIOD CURRENTED	PCED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x020	USBH_OHCI_ HCCONTROL HEADED	CHED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x024	USBH_OHCI_ HCCONTROL CURRENTED	CCED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x028	USBH_OHCI_ HCBULK HEADED	BHED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x02C	USBH_OHCI_ HCBULK CURRENTED	BCED[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x030	USBH_OHCI_ HCDONE HEAD	DH[27:0]																												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x034	USBH_OHCI_ HCFM INTERVAL	Res.	FSMPS[14:0]															Res.	Res.	FI[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			1	0	1	1	1	0	1	1	0	1	1	1	1	1
0x038	USBH_OHCI_ HCFM REMAINING	FRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FR[13:0]													
	Reset value	0																		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	USBH_OHCI_ HCFM NUMBER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FN[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	USBH_OHCI_ PERIODIC START	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	USBH_OHCI_ HCLS THRESHOLD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LST[13:0]													
	Reset value																			0	0	0	1	1	0	0	0	1	0	1	0	0	0

Table 474. USBH register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	**USBH_OHCI_ HCRH DESCRIPTOR A**	POTPGT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCP	OCPM	DT	NPS	PSM	NDP[7:0]							
	Reset value	0	0	0	0	0	0	1	0												0	0	0	0	0	0	0	0	0	0	0	1	0
0x04C	**USBH_OHCI_ HCRH DESCRIPTOR B**	PPCM[15:0]																DR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	**USBH_OHCI_ HCRH STATUS**	CRWE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCIC	LPSC	DRWE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCI	LPS
	Reset value	0														0	0	0														0	0
0x0054	**USBH_OHCI_ HCRHPORT STATUS1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRSC	OCIC	PSSC	PESC	CSC	Res.	Res.	Res.	Res.	Res.	Res.	LSDA	PPS	Res.	Res.	Res.	PRS	POCI	PSS	PES	CCS
	Reset value												0	0	0	0	0							0	0				0	0	0	0	0
0x0058	**USBH_OHCI_ HCRHPORT STATUS2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRSC	OCIC	PSSC	PESC	CSC	Res.	Res.	Res.	Res.	Res.	Res.	LSDA	PPS	Res.	Res.	Res.	PRS	POCI	PSS	PES	CCS
	Reset value												0	0	0	0	0							0	0				0	0	0	0	0
0x0098	**USBH_OHCI_ INSNREG06**	AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	H BURST [2:0]			EXP_BEATS [4:0]					SUC_ BEATS [3:0]			
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x009C	**USBH_OHCI_ INSNREG07**	AHB_MST_ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1000	**USBH_EHCI_ HCICAP LENGTH**	HCIVERSION[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPLENGTH[7:0]							
	Reset value	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0									0	0	0	1	0	0	0	0
0x1004	**USBH_EHCI_ HCSPARAMS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPN[3:0]				Res.	Res.	Res.	P_INDICATOR	N_CC[3:0]				N_PCC [3:0]				PRR	Res.	Res.	PPC	N_PCC [3:0]			
	Reset value									0	0	0	0				0	0	0	0	1	0	0	1	0	0			1	0	0	1	0
0x1008	**USBH_EHCI_ HCCPARAMS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPMC	Res.	EECP[7:0]								ISO[3:0]				Res.	ASPC	PFLF	SFAC
	Reset value															1		1	0	1	0	0	0	0	0	0	0	1	0		1	1	0
0x1010	**USBH_EHCI_ USBCMD**	Res.	Res.	Res.	Res.	HIRD[3:0]				ITC[7:0]								Res.	Res.	Res.	Res.	ASPME	Res.	ASP MC [1:0]		LHCR	IAAD	ASE	PSE	FLS[1:0]		HCR	RS
	Reset value					0	0	0	0	0	0	0	0	0	1	0	0					1		1	1	0	0	0	0	0	0	0	0

Table 474. USBH register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1014	USBH_EHCI_ USBSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASS	PSS	REC	HCH	Res.	Res.	Res.	Res.	Res.	Res.	IAA	HSE	FLR	PCD	USBERRINT	USBINT
	Reset value																	0	0	0	1							0	0	0	0	0	0
0x1018	USBH_EHCI_ USBINTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAAE	HSEE	FLRE	PCIE	USBERRINTE	USBINTE
	Reset value																											0	0	0	0	0	0
0x101C	USBH_EHCI_ FRINDEX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRINDEX[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1024	USBH_EHCI_ PERIODICLIST BASE	PERIODICLISTBASE[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x1028	USBH_EHCI_ ASYNCLIST ADDR	LPL[31:5]																											Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x1050	USBH_EHCI_ CONFIGFLAG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF
	Reset value																																0
0x1054	USBH_EHCI_ PORTSC1	DEVADDR[6:0]							SS[1:0]		WKOC_E	WKDSCNNT_E	WKCNNT_E	PTC[3:0]				PIC[1:0]		PO	PP	LS[1:0]		SUSP_L1	PR	SUSPEND	FPR	OCC	OCA	PEDC	PED	CSC	CCS
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1058	USBH_EHCI_ PORTSC2	DEVADDR[6:0]							SS[1:0]		WKOC_E	WKDSCNNT_E	WKCNNT_E	PTC[3:0]				PIC[1:0]		PO	PP	LS[1:0]		SUSP_L1	PR	SUSPEND	FPR	OCC	OCA	PEDC	PED	CSC	CCS
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1094	USBH_EHCI_ INSNREG01	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT_THRESHOLD [7:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN_THRESHOLD [7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0x1098	USBH_EHCI_ INSNREG02	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT_BUF[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 474. USBH register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x109C	**USBH_EHCI_ INSNREG03**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE_LS_GLITCH	Res.	EN_CLK256_CHECK	TESTSE_NAK	TX_TRA_DELAY[2:0]			FRM_LST_FETCH	TIME_AVAIL_OFF[7:0]								BRK_MEM_TRANS
	Reset value																1		0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0x10A0	**USBH_EHCI_ INSNREG04**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPEND_SIGNAL	Res.	Res.	Res.	HCCPARAMS_WR	HCSPARAMS_WRT
	Reset value																											0				0	0
0x10A4	**USBH_EHCI_ INSNREG05**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSY	VPORT [3:0]				VCONTROL_LDM	VCONTROL[3:0]				VSTATUS[7:0]							
	Reset value															0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x10A8	**USBH_EHCI_ INSNREG06**	AHB_ECAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HBURST[2:0]			EXP_BEATS[4:0]					SUC_BEATS[3:0]			
	Reset value	0																				0	0	0	0	0	0	0	0	0	0	0	0
0x10AC	**USBH_EHCI_ INSNREG07**	AHB_MST_ERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

61 Ethernet (ETH): gigabit media access control (GMAC) with DMA controller

61.1 Ethernet introduction

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications, including AV nodes.

61.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

61.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC, gigabit media independent interface (GMII) and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB slave port
- AMBA4 for AXI master port
- RGMII specification version 2.6 from HP/Marvell
- RMII specification version 1.2 from RMII consortium

61.2.2 MAC features

MAC Tx and Rx common features

- Separate transmission, reception, and control interfaces to the application
- 10, 100, and 1000 Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant GMII/MII (default) interface to communicate with an external gigabit or Fast Ethernet PHY
  - RGMII interface to communicate with an external gigabit PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using backpressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
  - Packet bursting and packet extension in 1000 Mbps Half-duplex operation

- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in (G)MII and reduced gigabit media independent interface (RGMII) PHYs
- 64-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
- Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
- Flexibility to control pulse-per-second (PPS) output signal (eth_ptp_pps_out and ETH_PPS_OUT)
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

MAC Tx features

- Preamble and start-of-frame data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in Full-duplex mode
- Insert, replace, or delete queue/channel-based VLAN tags

MAC Rx features

- Automatic Pad and CRC stripping options
- Option to disable automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - Four 48-bit perfect (DA) address filters with masks for each byte
  - Four 48-bit SA address comparison check with masks for each byte
  - 64 bit Hash filter for multicast and unicast (DA) addresses
- Option to pass all multicast addressed packets
- Promiscuous mode to pass all packets without any filtering for network monitoring
- Pass all incoming packets (as per filter) with a status report

- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Detection of remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in Full-duplex mode)
- Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN tags and providing the tags in the status

61.2.3 Transaction layer (MTL) features

MTL Tx and Rx Common Features

- 64-bit Transaction Layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features

- 4096-byte Transmit FIFO with programmable threshold capability
- Two Tx queues on the Transmit path with a common memory for all Tx queues
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable queue size in configurations with multiple queues. Each queue size can be programmed in terms of 256 bytes
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Optional statistics related to bandwidth consumption for each queue of up to 16 blocks over a 125 µs period
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues

  - Weighted round robin (WRR)
  - Strict priority (SP)
  - Credit-based shaper (CBS)

MTL Rx features

- 4096-byte Receive FIFO with configurable threshold
- Two Rx queues on the Receive path with a common memory for all Rx queues
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted round robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict priority (SP)

61.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see *Section 61.11.2: Ethernet DMA registers*) to control the DMA operations. The DMA block supports the following features:

- 64-bit data transfers
- Separate DMA channel in the Transmit path for each queue
- Single DMA channels for both queues in Receive path
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host control access and host data interface (AXI)
- Tx DMA channels with TCP segmentation offload (TSO) feature enabled

- Time-sensitive conditional packet transmission by comparing the slot time information provided in the descriptor (useful for AV applications)
- Programmable control for Transmit Descriptor posted writes to improve the throughput

61.2.5 Bus interface features

AXI master interface

The AXI master interface features are the following:

- Interfaces with the application through AXI4 compatible interface
- 32-bit address width
- 64-bit data
- OKAY, SLVERR, and DECERR responses
- Software-selected type of AXI burst (fixed and variable length burst) in AXI master interface; extended length fixed bursts of 32, 64, 128, and 256
- Handshaking on AXI read and write data channels

AHB slave interface

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry
- Error

61.2.6 Audio and video features

The Ethernet peripheral can be used in Audio Video (AV) mode. The supported features are compliant with the industry standards for AV traffic:

- Separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes
- Multiple Rx queues on the Receive paths for AV traffic, and multiple Tx queues on the Transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for Transmit channels
- Single Tx FIFO and Rx FIFO (MTL) for all selected queues
- Programmable Slot Interval with range from 1 µs to 4096 µs and granularity of 1 µs.

61.2.7 Generic queuing features

The Ethernet peripheral supports the following features for generic queuing:

- Programmable control for routing Receive packets with Multicast/Broadcast destination address to a programmable Receive queue
- Support routing of untagged Receive packets to a programmable Receive queue
- Programmable control for routing VLAN tagged and untagged IEEE 1588 PTP over Ethernet Receive packets to a same or separate programmable Receive queue
- Programmable control for routing Unicast/Multicast Receive packets that fail the destination address filter to a separate programmable Receive queues

61.3 Ethernet pins and internal signals

Table 475 lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII, GMII, RMII or RGMII) and on the device configuration.

Table 476 shows the internal Ethernet signals.

Table 475. Ethernet peripheral pins

Port name	Digital port type	Description
ETH_COL	Input	Collision detection signal, GMII/MII only.
ETH_CRS	Input	Carrier sense signal, GMII/MII only
ETH_REF_CLK	Input	RMII reference clock
ETH_RX_CLK	Input	MII/GMII timing reference for Rx data transfers
ETH_RXD[7:0]	Input	Receive data. 8 pins for GMII, 4 pins for MII and RGMII, 2 for RMII.
ETH_RX_DV	Input	Receive data valid
ETH_CRS_DV	Input	Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
ETH_RX_CTL	Input	After demultiplexing, this pin provides ETH_RX_DV on the rising edge of ETH_RX_CLK, and ETH_RX_ER (logical XOR between ETH_RX_CTL values on rising edge and falling edge of ETH_RX_CLK).
ETH_RX_ER	Input	Receive error
ETH_TX_CLK	Input	MII timing reference for Tx data transfers
ETH_GTX_CLK	Output	GMII/RGMII timing reference for Tx data transfers
ETH_TXD[7:0]	Output	Transmit data. 8 pins for GMII, 4 pins for MII and RGMII, 2 for RMII.
ETH_TX_EN	Output	Transmit data enable
ETH_TX_CTL	Output	Multiplexing of ETH_TX_EN on the rising edge of ETH_TX_CLK, and a logical XOR between ETH_TX_EN and ETH_TX_ER values on falling edge of ETH_TX_CLK

Table 475. Ethernet peripheral pins (continued)

Port name	Digital port type	Description
ETH_TX_ER	Output	Transmit error
ETH_MDC	Output	Management data clock
ETH_MDIO	Input/output	Management data
ETH_PHY_INTN	Input	PHY interrupt
ETH_PPS_OUT	Output	PTP pulse-per-second ETH_PPS_OUT
ETH_PTP_AUX_TS(1)	Input	Trigger input for the auxiliary snapshots of the PTP system time

1. This pin is not available on all devices.

Table 476. Ethernet internal input/output signals

Signal name	Signal type	Description
eth_hclk	Digital input	AHB clock
eth_aclk	Digital input	AXI clock
eth_sbd_intr_it	Digital output	Main Ethernet interrupt
lpi_intr_o	Digital output	Sideband signal generated when the transmitter or receiver enters or exits the LPI state.
pmt_intr_o	Digital output	Sideband signal generated when a valid remote wake-up packet is received
eth_tx_clk	Digital input	Tx kernel clock
eth_rx_clk	Digital input	Rx kernel clock
eth_rmii_ref_clk	Digital input	RMII reference kernel clock
eth_ptp_pps_out	Digital output	PTP pulse-per-second signal
mac_speed_o[1:0]	Digital output	MAC speed information used by the RCC
clk_ptp_ref_i	Digital input	PTP reference clock input
eth_ptp_trig[3:0]	Digital input	Trigger input for auxiliary snapshots of the PTP system time

61.4 Ethernet architecture

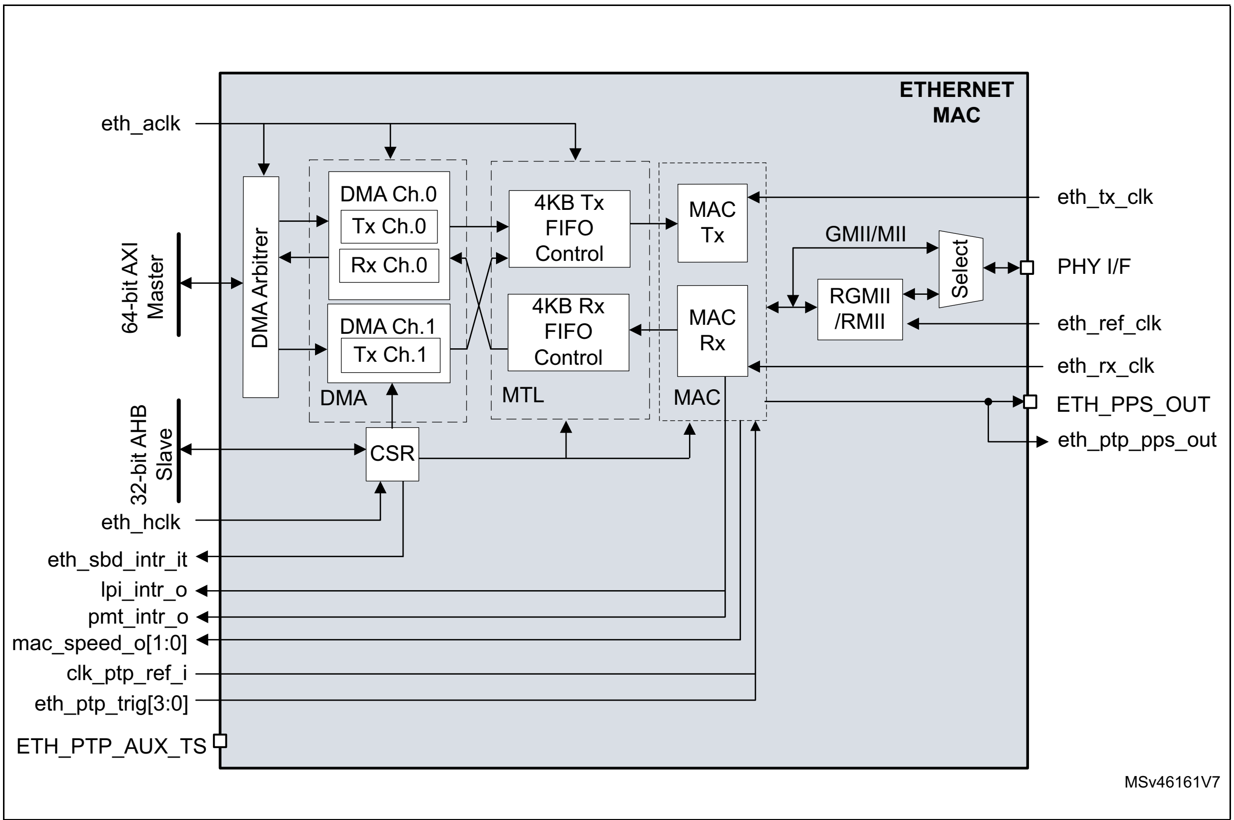
The Ethernet peripheral is composed of 4 main functional modules:

- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)

  This is the logical DMA module with one physical channel for reception and two for transmission. It controls the data transfers between MAC and system memory through the AMBA AXI4 64-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII or the RGMII PHY Media Independent Interfaces.

Figure 732. Ethernet high-level block diagram



1. For a definition of the internal signals, refer to *Table 476*.
2. Refer to RCC chapter "Clock distribution for Ethernet" for a detailed description of the Ethernet clock architecture.
3. The ETH_PTP_AUX_TS pin is not available on all devices.

61.4.1 DMA controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: *Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*) and *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*.

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*.

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

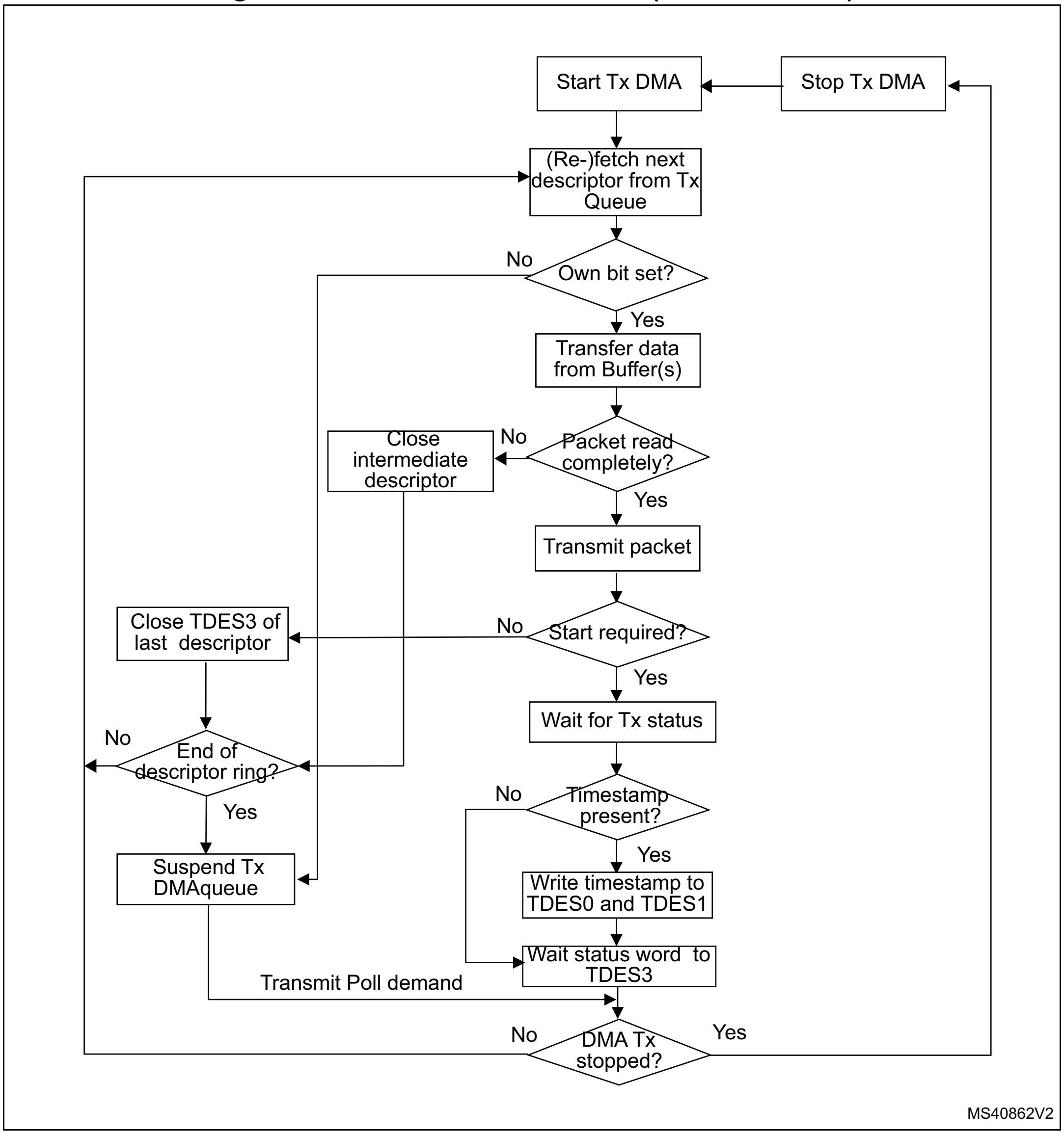
Descriptors are specified in *Section 61.10: Descriptors*.

DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the descriptor tail pointer offset value of the Transmit channel.
3. While in the Run state, the DMA runs an arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   – The descriptor is flagged as owned by the application (TDES3 [31] = 0).
   – The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   – An error condition occurs.
6. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
7. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
9. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.
10. Bit 0 of *Channel x status register (ETH_DMACxSR)*) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.

Figure 733. DMA transmission flow (standard mode)



DMA transmission in OSP (Operate on Second Packet) mode

In Run state, if bit 4 is set in the *Channel x transmit control register (ETH_DMACxTXCR)*, the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

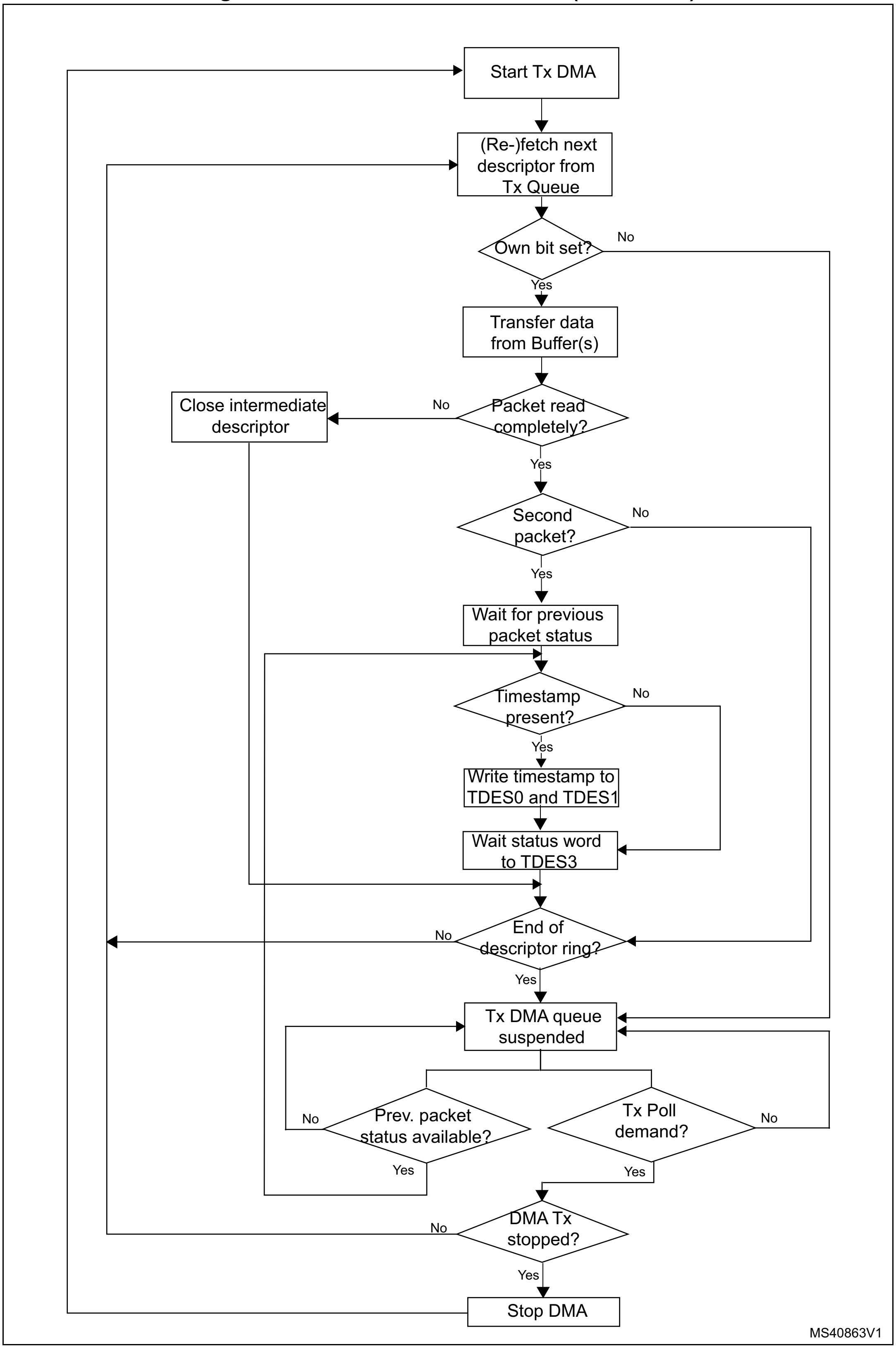
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see *Section : DMA transmission in default mode*).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.

   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in *Figure 735: Receive DMA flow*.

Figure 734. DMA transmission flow (OSP mode)



DMA reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

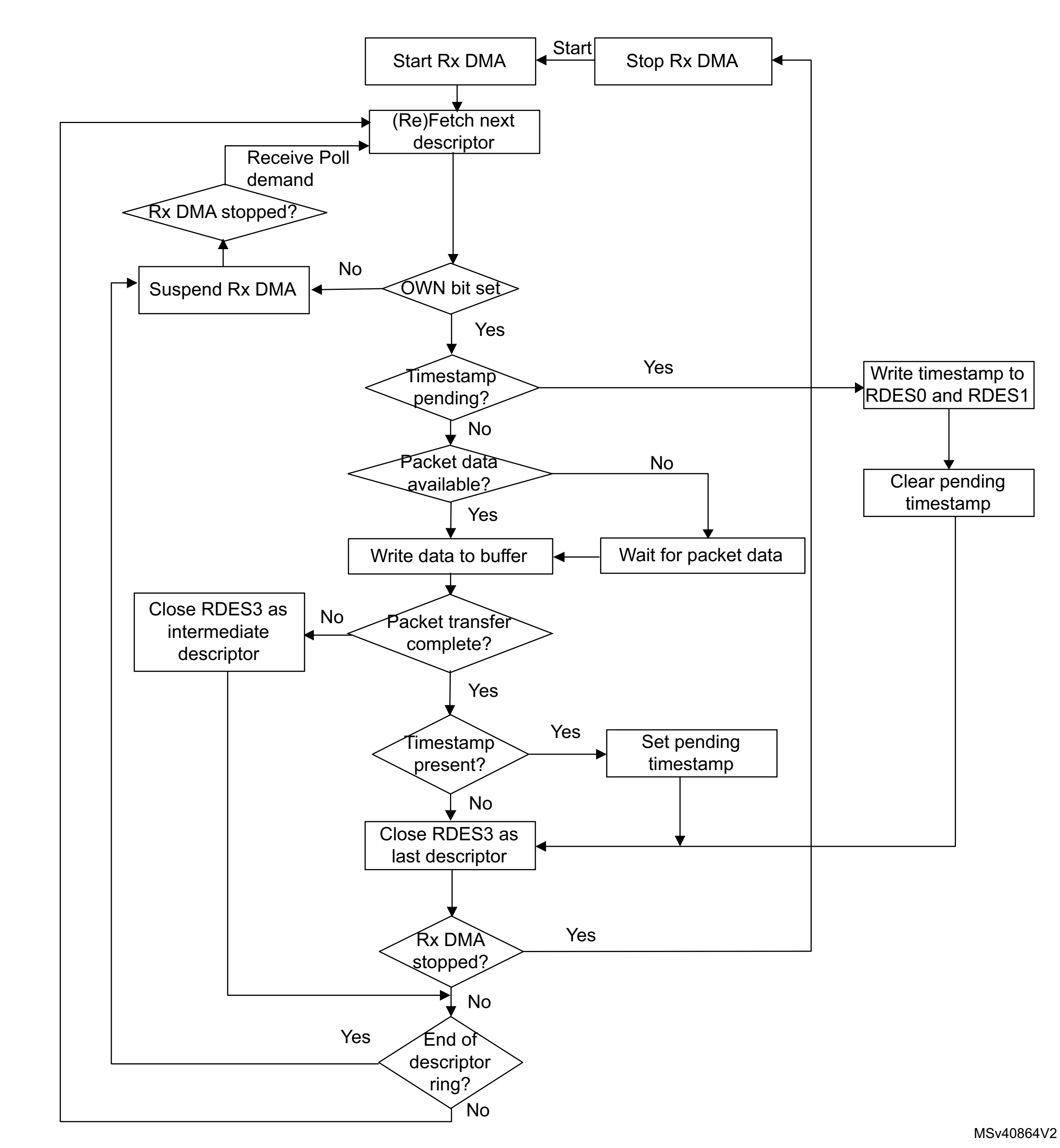
The DMA Rx descriptor ring structure is described in *Section 61.10: Descriptors*.

The reception sequence for Rx DMA engine is as follows (see also *Figure 735: Receive DMA flow*):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application should set the correct value in the Receive descriptor tail pointer register of corresponding DMA channel to indicate the location of the last valid descriptor for the DMA. If the tail pointer points to descriptor N, the last valid descriptor for the DMA is descriptor N - 1.
2. When bit 0 of *Channel 0 receive control register (ETH_DMAC0RXCR)*) is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. The descriptors referenced between the current descriptor and the tail pointer registers are available for the DMA. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA retrieves the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given, and the application updates the channel Receive descriptor tail pointer register to indicate the location of the last valid descriptor for DMA. Then, the engine proceeds to step 2 and fetches again the next descriptor.

Note: Refer to Section : Descriptor tail pointer handling examples for updating the correct value in receive descriptor tail pointer register.

Figure 735. Receive DMA flow



Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the *DMA mode register (ETH_DMAMR)*.

61.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 64-bit wide data paths: the Transmit path and the Receive Path.

- Transmit path

  The application or internal DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue in Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 4096 bytes.
- Receive path

  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured Receive threshold (RTC bits[1:0] defined in the operating mode register of the corresponding queue), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 4096 bytes.

61.4.3 MAC

The MAC is responsible of the Ethernet protocol processing.In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the GMII or MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   – The MAC completes the normal transmission and provides the transmission status to the MTL.
   – If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:

     the packet was successfully transmitted;

     the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.

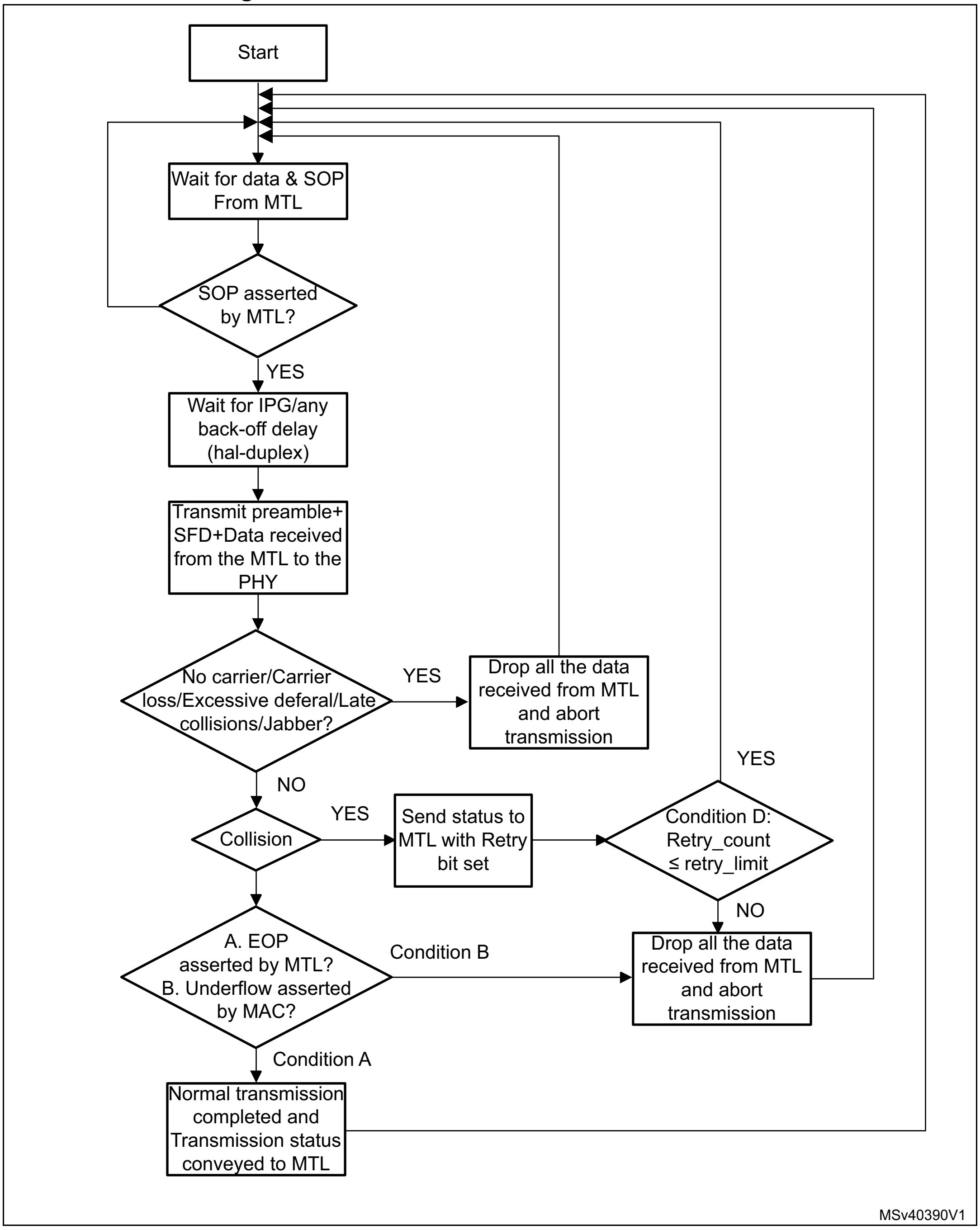
– If any one of the following event happens, the MAC aborts the packet transmission:
  - no carrier (Half-duplex mode)
  - loss of carrier (Half-duplex mode)
  - excessive deferral (Half-duplex mode)
  - late collisions (Half-duplex mode)
  - jabber

  the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.
5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

Figure 736: Overview of MAC transmission flow illustrates the MAC transmission process flow.

Figure 736. Overview of MAC transmission flow



MAC reception

A receive operation is initiated when the MAC detects an SFD on GMII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

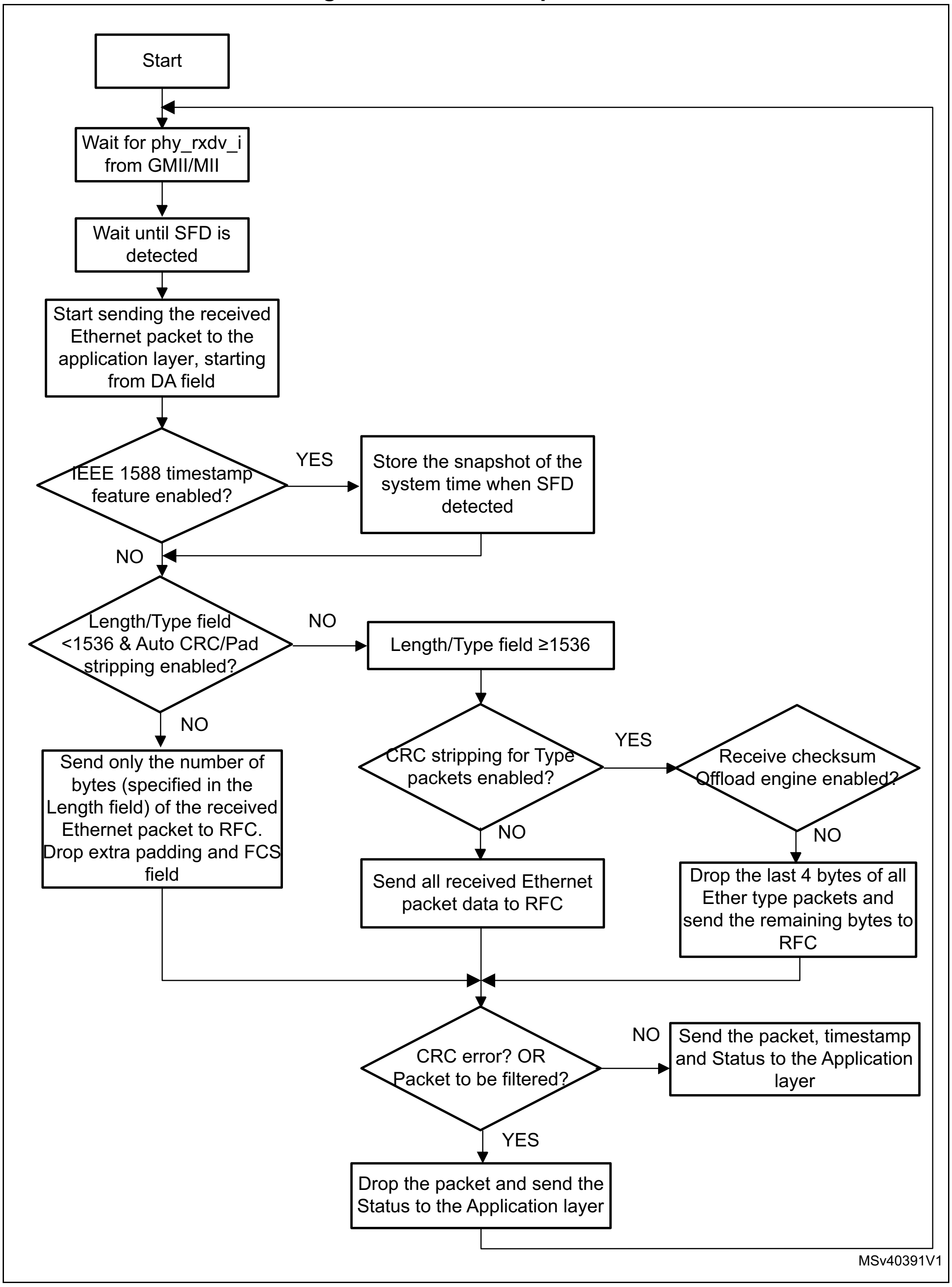
The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of GMII or MII becomes active, the Receive State Machine (RSM) starts looking for the SFD field (0x5D byte).

   The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).
3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on GMII or MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application.The receive state machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the *Operating mode configuration register (ETH_MACCR)*), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.
4. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the *Operating mode configuration register (ETH_MACCR)*. However, if the CRC stripping has been enabled for Type packets and not enabled the Receive Checksum Offload Engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.
5. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of *Watchdog timeout register (ETH_MACWTR)*) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of *Operating mode configuration register (ETH_MACCR)*. However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.

Figure 737. MAC reception flow



61.4.4 Multiple channels and queues support

This chapter describes how the Ethernet peripheral core supports multiple queues and channels.

Two queues on Rx side and two queues on Tx side are supported. Two DMA channels are implemented on Tx side, one channel per queue, while only one single DMA channel is implemented for both queues on Rx path.

The multiple queues enable to support the Audio video bridging (AVB) protocol.

Multiple queues and channels support in the Transmit path

The number of Tx DMA channels is equal to the number of Tx queues, and is direct one-to-one mapping. This enables the interleaving of packet/data transfer of multiple Tx DMA on the host/system bus, and each Tx queue is reserved for the corresponding Tx DMA. Therefore, the integrity of packet transfer by each Tx DMA is maintained in the Tx queue and enables efficient utilization of the host bus for data transfers among multiple Tx DMA without depending on the packet length, Transmit buffer availability and so on.

If two Tx DMAs are allowed to transfer data into the same Tx queue, when a Tx DMA starts a packet transfer, the other Tx DMA cannot transfer data to the same Tx queue unless the previous packet transfer is complete. This means that the second DMA remains idle until the first packet transferred is complete, which is effectively a one-to-one mapping of the Tx DMA to the Tx queue.

The Ethernet peripheral supports multiple Tx channels. The priority scheme between both channels is fixed priority. In fixed priority scheme, the channel with the highest priority always wins the arbitration when it requests the bus, as shown in *Table 477*.

Table 477. Fixed priority scheme for DMA channels

Priority level	Channel
0	Channel 0
1	Channel 1

The fixed priority scheme does not necessarily cause the “starving” of lower priority channels for the AXI bus because of the capabilities of the AXI protocol and characteristics of the TxDMA, described as follows:

- Read command requests from a TxDMA for a burst of data transfers are driven on a separate AXI channel, and their execution takes one clock cycle. The actual data transfer starts on a separate AXI channel only after the request is accepted. It takes several clock cycles to complete with relatively large read access latencies.
- The AXI master is free to issue subsequent requests even before the previous data transfers are complete.
- A TxDMA requests another data transfer only after the data transfer corresponding to its previous request is complete and transferred to the MTL Tx queue and space is available in Tx queue to accept the next request.
- The Ethernet peripheral can be programmed to control the maximum number of outstanding requests, requested burst sizes by DMA, and the burst sizes of each data transfer on the AXI bus.

Considering the previous points, the Ethernet peripheral can be programmed so that the requests of all TxDMAs are placed and executed on the AXI bus before the data transfer of the first TxDMA is complete and it raises the next request to the arbiter. For example, in a 64-bit AXI data bus, proceed as follows:

1. TxDMA PBL = 64 (512 bytes).
2. Number of outstanding AXI read requests = 16.
3. Maximum burst length on AXI (BLEN) = 32 (256 bytes).
4. Number of active TxDMAs = 8.

Assuming that all TxDMAs place a request simultaneously and get serviced by the fixed priority algorithm, all the requests are placed on the AXI bus in a window of 32 clocks (two clocks per AXI request, two AXI burst requests per TxDMA request). The data transfer of the first request takes 32 clock cycles + the read-access latencies (assuming that there are tens of cycles). So, the TxDMA arbiter is already IDLE and has completed servicing all the TxDMA requests before this data transfer completes.

Effectively, all the TxDMA channels get equal access to AXI bus and the corresponding Tx queues get filled up at the same rate. The data gets read out from the Tx queues as per the traffic class scheduler in the MTL. So at a steady state, the bandwidth for each channel is controlled by the priorities and settings of the traffic scheduler in the MTL.

Programming guidelines for multiple queues and channels in the Transmit path

See *Section 61.9.8: Programming guidelines for multichannel multiqueue operation on page 3284*.

Multiple queues and channels support in the Receive path

The Ethernet peripheral supports only one channel in Rx direction.

Programming guidelines for multiple queues and channels in the Receive path

See *Section 61.9.8: Programming guidelines for multichannel multiqueue operation on page 3284*.

Selection of the tag priorities assigned to Rx queues

The VLAN tag priorities can be assigned to Rx queues by programming the PSRQ field in the corresponding *Rx queue control 2 register (ETH_MACRXQC2R)*. The bit corresponding to the VLAN tag priority can be set in the PSRQ field to assign the corresponding priority to the Rx queue. As an example, to assign VLAN tag priority 3 to Rx queue 0, set bit3 of PSRQ field in the *Rx queue control 2 register (ETH_MACRXQC2R)*. The VLAN tag priority assigned to a given Rx queue must be unique. This means that multiple Rx queues cannot be assigned the same VLAN tag priority. However, multiple VLAN tag priorities can be assigned to the same Rx queue.

The settings in the PSRQ field is used for VLAN tagged Rx packet routing to Rx queues as well as for PFC based Tx flow control. The received VLAN tagged Rx packet is routed to the Rx queue that matches the VLAN tag priority. In PFC-based Tx flow control, the PSRQ field corresponding to a particular Rx queue is used to enable VLAN tag priorities in the PFC packet that is transmitted when the corresponding Rx queue threshold levels are reached.

Rx side routing from MAC to queues

The MAC routes the Rx packets to the Rx queues based on following packet types:

- Unicast/Multicast destination address packets that fail the destination address filter
- Multicast/Broadcast destination address packets that pass the destination address filter
- VLAN tag priority field in VLAN tagged AV data packets
- AV control packets
- VLAN Tagged IEEE 1588 PTP over Ethernet packets
- Untagged IEEE1588 PTP over Ethernet packets
- VLAN tag priority field in VLAN tagged packets
- Untagged packets

The outer C-VLAN tagged AV data Rx packets can be routed following the priorities assigned to Rx queues in PSRQ field of the corresponding *Rx queue control 2 register (ETH_MACRXQC2R)*. The corresponding Rx queue is enabled for AV through RXQxEN field in *Rx queue control 0 register (ETH_MACRXQC0R)*. These packets can be:

- single VLAN tagged with C-VLAN type
- or double VLAN tagged with outer VLAN tag of C-VLAN type when the double VLAN feature is enabled (EDVLP bit set in *VLAN tag register (ETH_MACVTR)*) with inner C-VLAN tagged, or inner S-VLAN tagged when SVLAN processing is enabled (ESVL bit set in *VLAN tag register (ETH_MACVTR)*).

The AV control Rx packets can be routed following the Rx queue number specified in the AVCPQ field of *Rx queue control 1 register (ETH_MACRXQC1R)*. The corresponding Rx queue is enabled for AV through RXQxEN field in *Rx queue control 0 register (ETH_MAC-RXQC0R)*. These packets can be:

- single VLAN tagged with C-VLAN type
- or double VLAN tagged with outer VLAN Tag of C-VLAN type when double VLAN feature is enabled (EDVLP bit set in *VLAN tag register (ETH_MACVTR)*) with inner C-VLAN tagged or inner S-VLAN tagged when SVLAN processing is enabled (ESVL bit set in *VLAN tag register (ETH_MACVTR)*).

The VLAN tagged Rx packets can be routed following the priorities assigned to Rx queues in PSRQ field of the corresponding *Rx queue control 2 register (ETH_MACRXQC2R)*. The corresponding Rx queue is enabled through RXQxEN field in the *Rx queue control 0 register (ETH_MACRXQC0R)*.

The untagged IEEE1588 PTP over Ethernet Rx packets can be routed following the Rx queue number specified in the PTPQ field in *Rx queue control 1 register (ETH_MACRX-QC1R)*. The corresponding Rx queue is enabled for AV through RXQxEN field in the *Rx queue control 1 register (ETH_MACRXQC1R)*.

The VLAN tagged IEEE 1588 PTP over Ethernet Rx packets can be routed based on the priorities assigned to RX queues through PSRQ[7:0] bitfields in *Rx queue control 2 register (ETH_MACRXQC2R)* or on the Rx queue number specified in the PTPQ[2:0] bitfield in *Rx queue control 1 register (ETH_MACRXQC1R)*. The corresponding Rx queue is enabled through RXQxEN[1:0] bitfield in *Rx queue control 0 register (ETH_MACRXQC0R)*. This is determined by programming in TPQC[1:0] bitfield of *Rx queue control 1 register (ETH_MACRXQC1R)*. The VLAN Tagged IEEE 1588 PTP over Ethernet Rx packets are detected only when 802.1AS mode is disabled (AV8021ASMEN bit in *Timestamp control Register (ETH_MACTSCR)* is set to 0), otherwise this type of Rx packets are routed as generic VLAN Tagged Rx packets.

The multicast/broadcast destination address Rx packets that pass the destination address filter can be routed based on the Rx queue number specified in the MCBCQ[2:0] bitfield of *Rx queue control 1 register (ETH_MACRXQC1R)* when enabled through MCBCQEN bit of *Rx queue control 1 register (ETH_MACRXQC1R)*. The corresponding Rx queue is enabled through RXQxEN[1:0] bitfield in *Rx queue control 0 register (ETH_MACRXQC0R)*.

The untagged Rx packets can be routed based on the Rx queue number specified in the UPQ[2:0] field of *Rx queue control 1 register (ETH_MACRXQC1R)*.The corresponding Rx queue is enabled through RXQxEN[1:0] bitfield in *Rx queue control 0 register (ETH_MACRXQC0R)*.

The unicast destination address Rx packets that fail the destination address filter can be routed based on the Rx queue number specified in the UFFQ field of *Rx Queue control register (ETH_MACRXQCR)* when enabled through UFFQE bit of *Rx Queue control register (ETH_MACRXQCR)* and RA bit of *Packet filtering control register (ETH_MACPFR)* is set. The corresponding Rx queue is enabled through RXQxEN[1:0] bitfield in *Rx queue control 0 register (ETH_MACRXQC0R)*.

The multicast destination address Rx packets that fail the destination address filter can be routed based on the Rx queue number specified in the MFFQ field of *Rx Queue control register (ETH_MACRXQCR)* when enabled through MFFQE bit of *Rx Queue control register (ETH_MACRXQCR)*, RA bit of *Packet filtering control register (ETH_MACPFR)* is 1.The corresponding Rx queue is enabled through RXQxEN[1:0] bitfield in *Rx queue control 0 register (ETH_MACRXQC0R)*.

If the Rx packet cannot be classified in any of the defined packet type for routing, it is routed through Rx queue 0.

Rx side arbitration between DMA and MTL

After completing the current packet processing, the DMA channel controller fetches the next descriptor. When the descriptor fetching is complete, the DMA channel controller evaluates the amount of data to be transferred to the Rx buffer based on the programmed PBL and Rx buffer length, and requests the MTL to transfer the data accordingly.

After servicing the current request, the MTL Rx queue arbitration scheme selects a Rx queue based on the arbitration scheme (RAA field of the *Operating mode register (ETH_MTLOMR)*) and the weights programmed in the *Rx queue x control register (ETH_MTLRXQxCR)*). The arbitration is done between queues for which DMA is ready to service. After the Rx queue is selected, PBL (Programmable Burst Length) amount of data is read out from this queue and routed to the Rx DMA channel depending on the Rx channel selected.

The arbitration takes place when every PBL data transfer is completed and descriptors are ready for the processing from at least one DMA channel.

AV feature

The Ethernet peripheral supports the AV data transfer in 100-Mbps and 1000-Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: enables the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.

- IEEE 802.1Qat-2009: enables the network resources to be reserved for specific traffic streams traversing a bridged local area network.
- IEEE 802.1AS-2011: specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 Full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

Transmit path functions

When the AV feature is selected, the Transmit paths of Tx queues are enabled by default.

The Transmit path of queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programmed threshold limit.

The Transmit path of the AVB Tx queue supports traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets.
- The credit for the queue is positive as per the algorithm.

When the credit-based shaper algorithm is disabled for a queue, the channel uses the default strict priority algorithm.

Each Transmit DMA has a separate descriptor chain for fetching the transmit data. The Transmit channel that gets the access to the system bus depends on the DMA arbiter.

The Transmit path has a shared FIFO (MTL layer) for each queue. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for queue 1, the queue is selected if the packet is available in the channel and has a positive or zero credit.

If the credit-based shaper algorithm is disabled for a given Tx queue, the packet to be transmitted is selected based on the fixed priority scheme as described in *Table 478*.

Table 478. Weight for DMA channels

TCW field	Transmit channel weight
000	1
001	2

Receive path functions

When the AV feature is enabled, the Receive path of Queue 0 is enabled by default. The whole traffic is received on this channel. The Receive paths of additional queues can be enabled. By enabling the Receive paths of multiple channels, the received data can be demultiplexed to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if the received packet is an AV packet as well as the corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the Receive descriptor as explained in *Section 61.10.4: Receive descriptor*. All received packets with an EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets

  AV data packets are always tagged. Tagged AV data packets are received following the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits[15:8] in the transmit flow control register of the corresponding queue.
- AV control packets

  The AV control packets can be either tagged or untagged. By default, untagged AV control packets are received on queue 0. To receive these packets on queue 1, program bits[2:0] of the *Rx queue control 1 register (ETH_MACRXQC1R)*. Similarly to the AV data packets, tagged AV control packets are received following the programmed priority value.

In addition to the AV packets, you can receive the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on queue 0. To receive these packets on any other queue, program bits[6:4] of the *Rx queue control 1 register (ETH_MACRXQC1R)*.

Credit-based shaper algorithm

The Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in queue 0. Queue 1 can be programmed to use the credit-based shaper algorithm.

The following sections provide information on the credit-based shaper algorithm implementation:

- Credit value
- idleSlopeCredit and sendSlopeCredit values
- Bandwidth status

Credit value

The credit value is accumulated every transmit clock cycle, that is every 40 ns in 100-Mbps mode and every 8 ns in 1000-Mbps mode. The credit to be added or subtracted per cycle can be fractional depending on the required idleSlope and sendSlope values, as described in *Table 479*.

Table 479. Example of credit value per transmit cycle

Mode	portTransmitRate, idleSlope, sendSlope	Credit value
100 Mbps	– portTransmitRate = 100 Mbps – idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) – sendSlope = 30 Mbps	– credit = 2.8 bits, accumulated per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. – credit = 1.2 bits, drained per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO depending on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA application bus.

The credit for a queue builds up only when the packet is available, but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet peripheral supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. This mode can be enabled through the CC bit of the CBS control register (*Tx queue 1 ETS control register (ETH_MTLTXQ1ECR)*.

- When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting.
- When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024 (such as 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229). In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x 1024, respectively.

Bandwidth status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register (*Tx queue x ETS status register (ETH_MTLTXQxESR)*) gives the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in bits[6:4] of the CBS control register of the respective queue. For example, if you have programmed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, queue 1 and queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and queue 0 uses the strict-priority algorithm. If queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

Slot number function

When the AV feature is enabled, the slot number function can be used to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.

The slot number at which the DMA should fetch the data from system memory can be programmed in the Transmit Descriptor Word 3 (TDES3). This 4-bit field enables the application to schedule data fetching at up to 16 slots of 125 µs each. This field is applicable only to the AV channels.

When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated every 125 µs of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, when the subsecond counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state until a match is found.

To enable the DMA to fetch data only if it matches the current slot or the next two slots, program bit 1 of the *Channel x slot function control status register (ETH_DMACxSFCSR)* of the corresponding DMA channel.

If the slot number in the descriptor is less than the reference slot number, the DMA considers it as a future slot.

The slot number check can be enabled by setting bit 0 in the *Channel x slot function control status register (ETH_DMACxSFCSR)* of corresponding DMA channel. If this check is disabled, the packets are fetched immediately after the descriptor is read. In addition, bits [19:16] indicate the value of the reference slot number in DMA.

Programming guidelines for AV feature

See *Section : Initializing the DMA on page 3292.*

See *Section : Enabling slot number checking on page 3293.*

See *Section : Enabling average bits per slot reporting on page 3293.*

Queue modes

A Tx queue can be enabled for generic or AV traffic. When multiple Tx queues are enabled for generic traffic, queuing is based on WRR or WSP algorithms.

When a Tx queue is enabled for AVB traffic, queuing is based on CBS or SP algorithms.

A Rx queue can be enabled for generic or AV traffic. A particular queue enabled for generic or AV based routing is determined by the RXQ0EN and RXQ1EN fields in the *Rx queue control 0 register (ETH_MACRXQC0R)*.

Below an explanation on how Rx queues are enabled based on the selected features:

- Multiple Rx queues

When multiple Rx queues are selected, all queues are enabled for generic queuing based on the VLAN tag priority. The VLAN tag priority should match the PSRQ[7:0] bitfield of the *Rx queue control 2 register (ETH_MACRXQC2R)*. By default untagged packets are routed to Receive queue specified in UPQ field of *Rx queue control 1 register*

(ETH_MACRXQC1R). Queue 0 is the default value of UPQ[2:0] bitfield. The default value can be overridden with any other value, for the UPQ[2:0] bitfield. The Rx packets can also be routed to a particular DMA channel based on the DCS field of the perfectly-matched MAC Address register (*MAC Address x high register (ETH_MACAxHR)*.

- Multiple Rx queues with AV feature

When AV is enabled for all selected Rx queues, the queuing is done based on the VLAN tag priority. The VLAN tag priority should match the PSRQ[7:0] bitfield of the *Rx queue control 2 register (ETH_MACRXQC2R)*. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register (*MAC Address x high register (ETH_MACAxHR)*). The AV control packets (tagged or untagged) are routed based on the AVCPQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*. The PTP over Ethernet packets are routed based on the PTPQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*.

61.5 Ethernet functional description: MAC

61.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

Transmit path

Table 480: Double VLAN processing features in Tx path describes the features supported by the MAC on the Transmit side.

Table 480. Double VLAN processing features in Tx path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN Tag deletion	VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word.

Receive path

Table 481: Double VLAN processing in Rx path describes the features supported by the MAC on the Receive side and the corresponding bits in the *VLAN tag register (ETH_MACVTR)*.

Table 481. Double VLAN processing in Rx path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.

Table 481. Double VLAN processing in Rx path (continued)

Feature	Description
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

61.5.2 Source address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets

  To enable this feature for all packets, program the SARC field of the *Operating mode configuration register (ETH_MACCR)*.
- Enabling SA insertion or replacement for selective packets

  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of Transmit Descriptor Word 3/TDES3, refer to *Section 61.10.3: Transmit descriptor*) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

VLAN insertion, replacement, or deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

  Insertion or replacement is performed based on the setting of VLTI bit in the *VLAN inclusion register (ETH_MACVIR)* as described in *Table 482: VLAN insertion or replacement based on VLTI bit*.

Table 482. VLAN insertion or replacement based on VLTI bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with the VLT field of *VLAN inclusion register (ETH_MACVIR)*

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of *VLAN inclusion register (ETH_MACVIR)*.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see *Table 515: TDES2 normal descriptor (read format)*).

  In addition, the VLP (VLAN Priority control) bit must be reset in *VLAN inclusion register (ETH_MACVIR)* (for outer VLAN) and *Inner VLAN inclusion register (ETH_MACIVIR)* (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

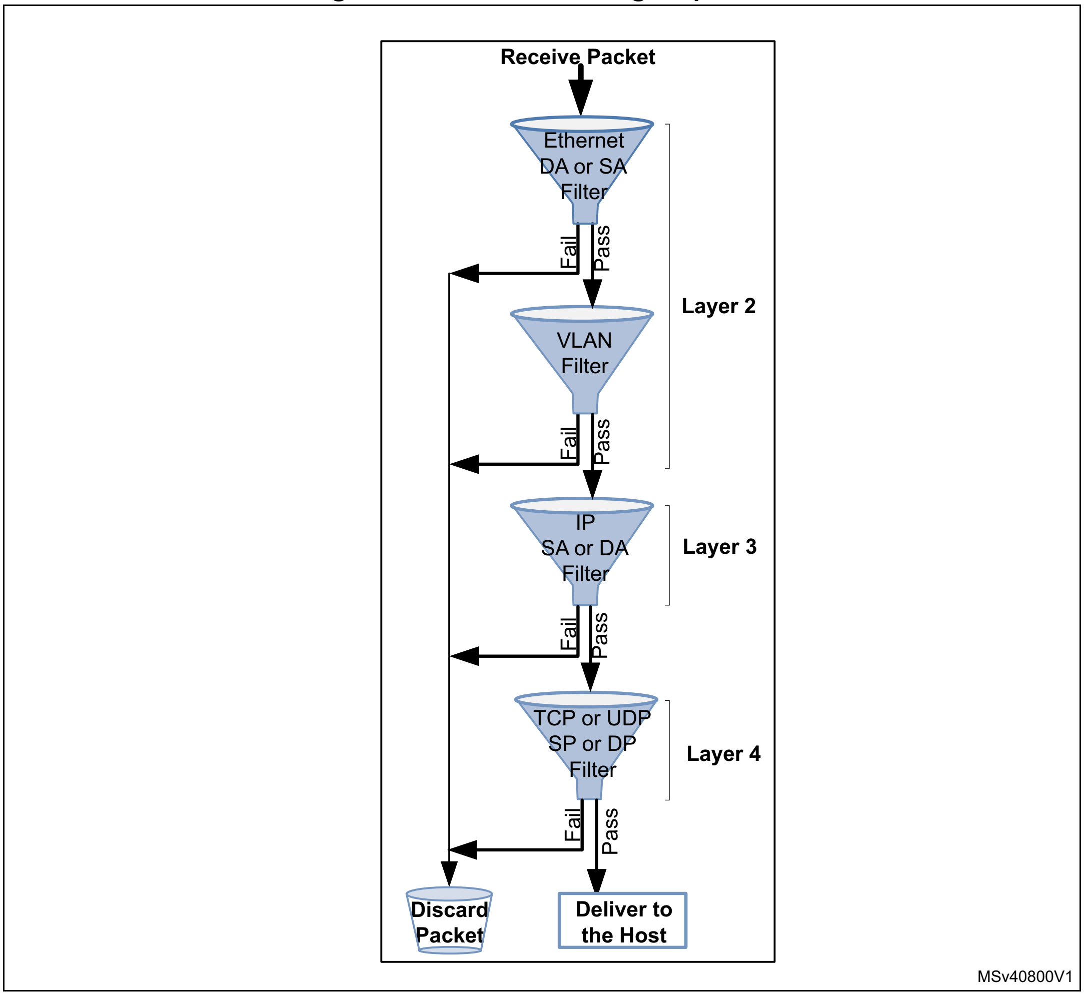
61.5.3 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- *MAC source or destination address filtering*: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- *VLAN filtering*: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- *Layer 3 and Layer 4 filtering*: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 738* shows the filtering sequence for Rx packets.

Figure 738. Packet filtering sequence



The sequence shown in *Figure 738* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of *Packet filtering control register (ETH_MACPFR)* control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

MAC source or destination address filtering

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast destination address filtering

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of *Packet filtering control register (ETH_MACPFR)* is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in *MAC Address x high register (ETH_MACAxHR)*. This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

Multicast destination address filtering

To program the MAC to pass all multicast packets, set the PM bit in *Packet filtering control register (ETH_MACPFR)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the *Packet filtering control register (ETH_MACPFR)*.

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

Hash or Perfect address filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Packet filtering control register (ETH_MACPFR)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

Broadcast address filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in *Packet filtering control register (ETH_MACPFR)*.

Unicast source address filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of *MAC Address x high register (ETH_MACAxHR)*.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in *Packet filtering control register (ETH_MACPFR)*. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see *Table 484*). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of *Packet filtering control register (ETH_MACPFR)*. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 483 and *Table 484* summarize the DA and SA filtering based on the type of packets received.

Note: When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 483. Destination address filtering

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
Multicast	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 484. Source address filtering

Packet type	PR	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to *Section 61.9.17: Programming guidelines to perform VLAN filtering on the receiver* for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of *VLAN tag register (ETH_MACVTR)*. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of *VLAN tag register (ETH_MACVTR)*. The VLAN packet status bit(bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

VLAN tag Hash filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the *VLAN tag register (ETH_MACVTR)*. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag register (ETH_MACVTR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 485 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of *Packet filtering control register (ETH_MACPFR)* is set, all packets are received and the VLAN match status is indicated in the VF bit of *RDES2 normal descriptor (write-back format)*. When the RA bit is not set and the VTFE bit is set in *Packet filtering control register (ETH_MACPFR)*, the packet is dropped if the final VLAN match status is Fail. In *Table 485*, value *X* means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of *VLAN tag register (ETH_MACVTR)*, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in *VLAN tag register (ETH_MACVTR)*.

Table 485. VLAN match status(1)

VID	VLAN perfect filter match result	VTHM Bit	VLAN Hash filter match result	VTIM bit	Final VLAN match status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, ‘X’ represents any value.

Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the *Operating mode configuration register (ETH_MACCR)* to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets

The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of *Operating mode configuration register (ETH_MACCR)* is set and one of the following conditions is true:

– All enabled Layer 3 and Layer 4 fields match.

– At least one of the enabled field matches and other fields are bypassed or disabled

When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets

  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value. Such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

- Non-TCP or UDP IP Packets

  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- *Layer4 Address filter 0 register (ETH_MACL4A0R)*
- *Layer3 Address 0 filter 0 register (ETH_MACL3A00R)*
- *Layer3 Address 1 filter 0 register (ETH_MACL3A10R)*
- *Layer3 Address 2 filter 0 register (ETH_MACL3A20R)*
- *Layer3 Address 3 filter 0 register (ETH_MACL3A30R)*

The second, and independent set of registers are: *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, *Layer 4 address filter 1 register (ETH_MACL4A1R)*, *Layer3 address 0 filter 1 Register (ETH_MACL3A01R)*, *Layer3 address 1 filter 1 register (ETH_MACL3A11R)*, *Layer3 address 2 filter 1 Register (ETH_MACL3A21R)* and *Layer3 address 3 filter 1 register (ETH_MACL3A31R)*.

61.5.4 IEEE 1588 timestamp support

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- *IEEE 1588 timestamp support*
- *IEEE 1588 system time source*
- *IEEE 1588 auxiliary snapshots*
- *Flexible pulse-per-second output*
- *PTP timestamp offload function*
- *One-step timestamp*"

IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588 2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

  The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

  – Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control Register (ETH_MACTSCR)*.

  – Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

  Table 486. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

  For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control Register (ETH_MACTSCR)*.

- Boundary clock

  The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

– The clock data sets are common to all ports of the boundary clock.

– The local clock is common to all ports of the boundary clock.

- End-to-end transparent clock

  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in *Table 487*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control Register (ETH_MACTSCR)*.

Table 487. End-to-end transparent clock: PTP messages for snapshot

PTP messages
SYNC
Delay_Req

- Peer-to-peer transparent clock

  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in *Table 495*.

Table 488. Peer-to-peer transparent clock: PTP messages for snapshot

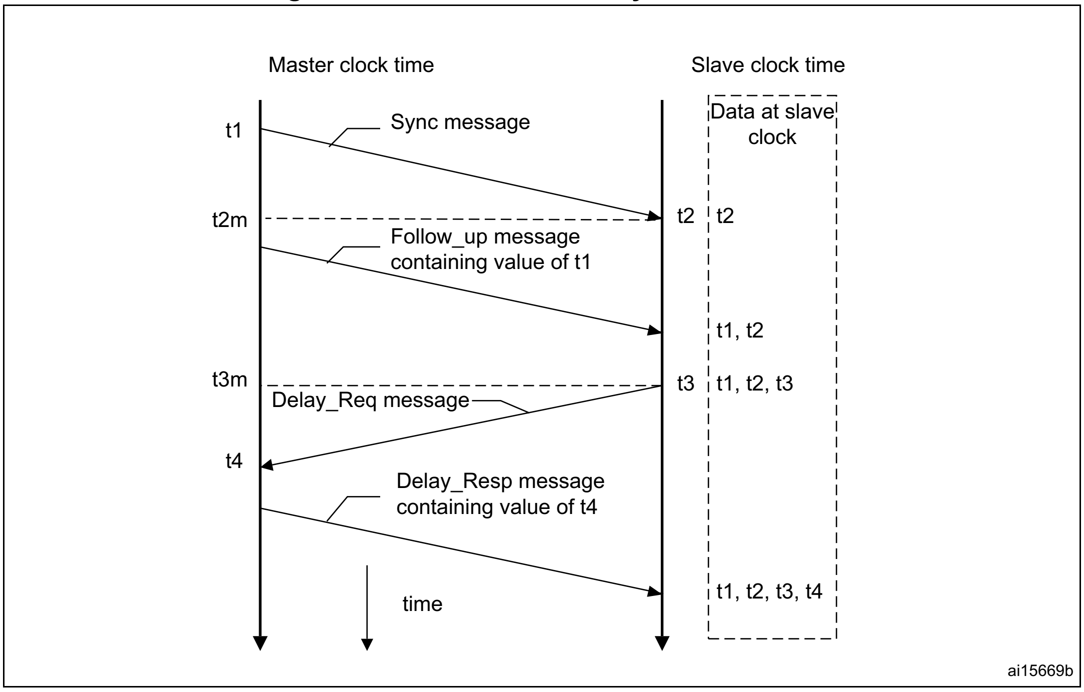
PTP messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in *Timestamp control Register (ETH_MACTSCR)*.

Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 739* shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

Figure 739. Networked time synchronization



As shown in *Figure 739*, the PTP uses the following process:

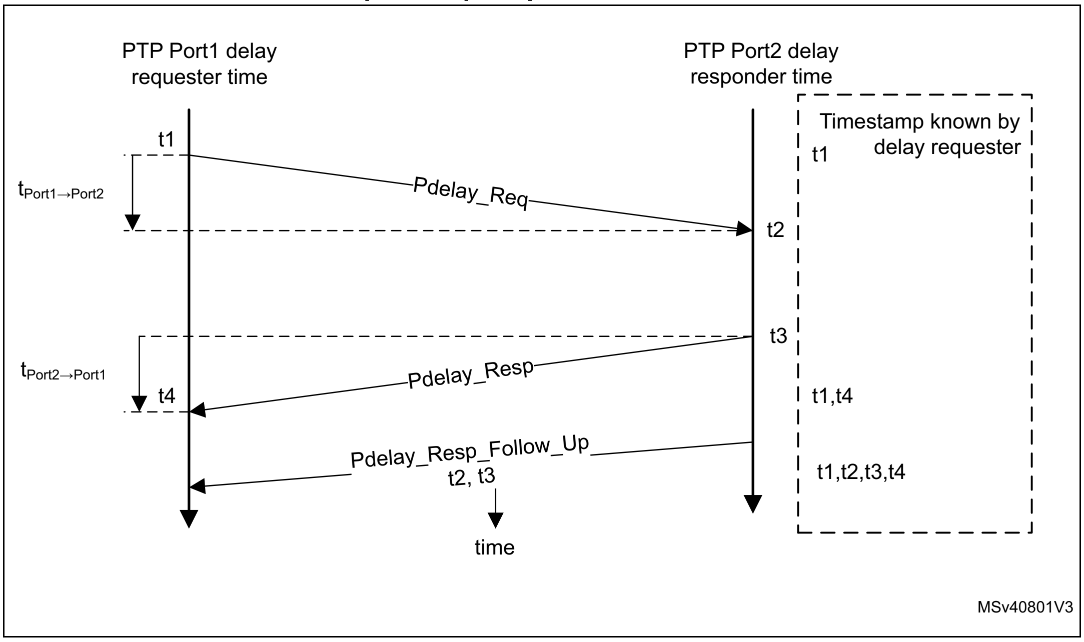
1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t_1. This time must be captured for Ethernet ports at GMII or MII.
2. The slave receives the SYNC message and also captures the exact time, t_2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, t_3, at which this packet leaves the GMII or MII interface.
5. The master receives the message, capturing the exact time t_4, at which the message enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the GMII or MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow_up, and Delay Response messages. *Figure 740* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 740. Propagation delay calculation in clocks supporting peer-to-peer path correction



As shown in *Figure 740*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t_1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t_2) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t_3) for this message.

   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:
   - Difference between the timestamps t_2 and t_3 in the Pdelay_Resp message
   - Difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message
   - Timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively
4. Port 1 generates a timestamp (t_4) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

Timestamp correction

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (GMII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the GMII and PTP clocks.

Ingress correction

In the Receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.
2. Internal latency from the input of the core to the internal capture point

   The latency differs based on the active PHY interface (such as MII or RMII) and the operating speed, as shown in *Table 489*.
3. CDC synchronization

   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).

The values determined from these three components should be added by the software and must be written into the TSIC field of the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

Note: The value written to the register must be negative (two's complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with “10^9 – total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is –5 ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of

~0.466 ns. Therefore, bits[30:0] must be written with "2^{31} – total ingress_correction_value" represented in binary with bit[31] = 1.

Egress correction

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp egress correction nanosecond register (ETH_MACTSECNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.
2. Internal latency from the internal capture point and the output of the core

   The latency differs based on the active PHY interface (RMII, MII, etc.) and the operating speed as shown in *Table 489*.
3. CDC synchronization error

   The timestamp correction because of synchronization is compensated by adding

   $$\text{EGRESS\_SYNC\_CORR} = (1 \times \text{PTP\_CLK\_PER} + 4 \times \text{TX\_CLK\_PER})$$

Table 489 lists the egress and ingress latency values for various PHY interfaces:

Table 489. Egress and ingress latency for PHY interfaces

PHY interface		Egress latency	Ingress latency
RGMII	1 Gbps	12	12
RGMII	100 Mbps	40	40
RGMII	10 Mbps	400	400
RMII	100 Mbps	40	120
RMII	10 Mbps	400	800

Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of GMII or MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- Maximum PTP clock frequency

  The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- Minimum PTP clock frequency

  The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the GMII or MII clock domain. This relationship is given by the following equation:

  3 × PTP clock period + 4 × GMII/MII clock period ≤ Minimum gap between two SFDs

  The GMII or MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 490*.

Table 490. Minimum PTP clock frequency example

Mode	Minimum gap between two SFDs	Minimum PTP frequency with internal timestamp
10 Mbps Full-duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
10 Mbps Half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
100 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
100 Mbps Half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
1000 Mbps Full-duplex	84 GMII clocks (64 clocks for a 64-byte packet + 12 clocks of min IFG + 8 clocks of preamble)	5 MHz
1000 Mbps Half-duplex	24 GMII clocks (4 clocks for a JAM pattern sent just after SFD because of collision + 12 IFG + 8 preamble)	18.75 MHz

PTP processing and control

Table 491 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

Table 491. Message format defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1

Table 491. Message format defined in IEEE 1588-2008 (continued)

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

PTP packets over IPv4

Table 492 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 491*.

Table 492. Message format defined in IEEE 1588-2008

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x0800	IPv4 datagram
IP version and header length	14	0x45	IP version is IPv4
Layer 4 protocol	23	0x11	UDP
IP multicast address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP multicast address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107

Table 492. Message format defined in IEEE 1588-2008 (continued)

Matched field	Octet position	Matched value	Description
UDP destination port	36, 37	0x013F, 0x0140	0x013F: PTP event messages(1) 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

PTP frames over IPv6

Table 493 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 491*.

Table 493. IPv6-UDP PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x86DD	IP datagram
IP version	14 (bits [7:4])	0x6	IP version is IPv6
Layer 4 protocol	20(1)	0x11	UDP
PTP multicast address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP destination port	56, 57a	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages

Table 493. IPv6-UDP PTP packet fields required for control and status (continued)

Matched field	Octet position	Matched value	Description
PTP control field (IEEE 1588 version 1)	94a	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP message type field (IEEE 1588 version 2)	62a (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension header is not defined for PTP packets.

PTP packets over Ethernet

Table 494 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 494. Ethernet PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC destination multicast address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC packet type	12, 13	0x88F7	PTP Ethernet packet
PTP control field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management

Table 494. Ethernet PTP packet fields required for control and status (continued)

Matched field	Octet position	Matched value	Description
PTP message type field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of *Timestamp control Register (ETH_MACTSCR)* is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the GMII or MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the Transmit Descriptor (see *Section 61.10.3: Transmit descriptor*). The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the GMII or MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the *Timestamp control Register (ETH_MACTSCR)*:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)

- Enable the node to be a master or slave and select the snapshot type

  This feature controls the type of messages for which snapshots are taken.

Note: The peripheral also supports the PTP messages over VLAN packets.

Table 495 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in Timestamp control Register (ETH_MACTSCR).

Table 495. Timestamp snapshot dependency on ETH_MACTSCR bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

The DMA returns the timestamp to the software application inside the corresponding Receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Programming guidelines for IEEE 1588 timestamping (system time correction)

See Section : System time correction in Section 61.9.10: Programming guidelines for IEEE 1588 timestamping on page 3287.

IEEE 1588 system time source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

Description of IEEE 1588 system time source

The peripheral uses the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:

- UInteger48 seconds field

  The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000 0000 0002.
- UInteger32 nanosecondsField

  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.

The nanoseconds field supports the following two modes:

– **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.
– **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is ~0.466 ns per bit.

  These modes can be set through TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)*.

System time register module

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the GMII.

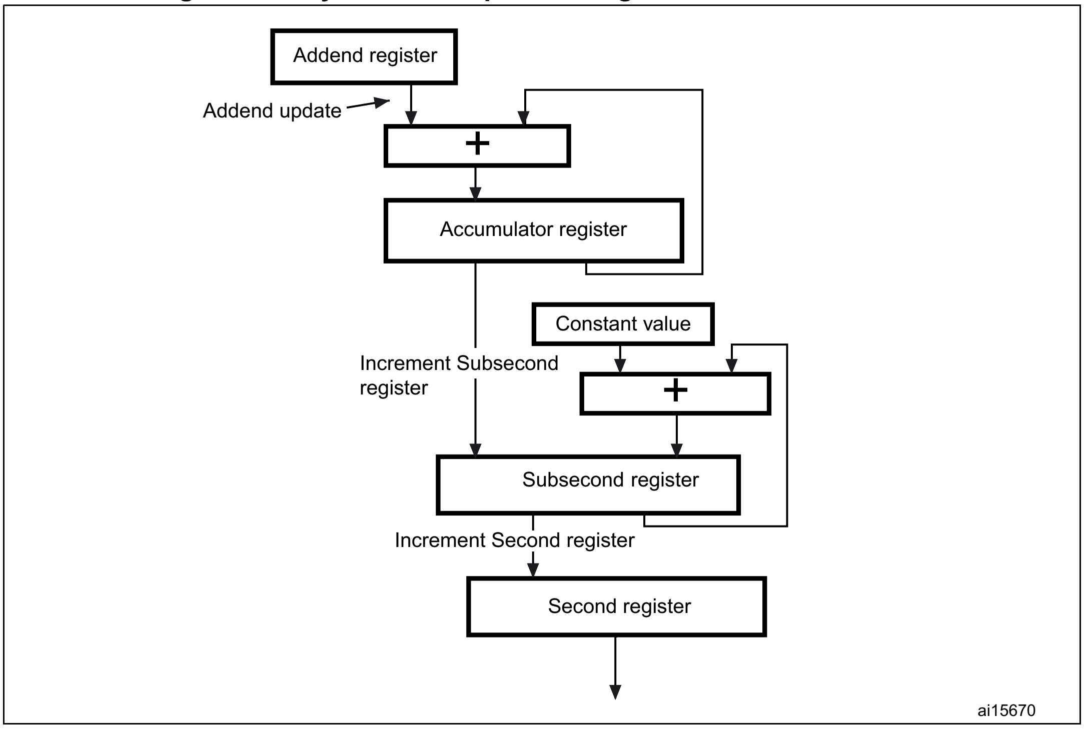
The system time counter can be initialized or corrected using either the coarse or the fine correction method.

In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 741*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 741*.

Figure 741. System time update using fine correction method



The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC 4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B 7672. When there is not clock drift, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In *Figure 741*, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\mathrm{FreqCompensationValue}_0 = 2^{32} / \mathrm{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

1. At time $MasterSyncTime_n$ the master sends the slave clock a SYNC message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as follows:

$$MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$$

2. The master clock counts for current Sync cycle, $MasterClockCount_n$ is

$$MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$$

(assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, $SlaveClockCount_n$ is

$$SlaveClockCount_n = SlaveClockTime_n - SlaveClockTime_{n-1}$$

4. The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is

$$ClockDiffCount_n = MasterClockTime_n - SlaveClockTime_n$$

5. The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is

$$FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$$

6. The frequency compensation value for Addend register, $FreqCompensationValue_n$ is

$$FreqCompensationValue_n = FreqScaleFactor_n \times FreqCompensationValue_{n-1}$$

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to *Section 61.9.10: Programming guidelines for IEEE 1588 timestamping* for detailed programming steps.

IEEE 1588 auxiliary snapshots

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where x = 1 to 4) sideband signal.

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: *Auxiliary timestamp seconds register (ETH_MACATSSR)* and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the *Timestamp status register (ETH_MACTSSR)* to know the timestamp of which input is available for reading at the top of this FIFO.

When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the *Timestamp status register (ETH_MACTSSR)*. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in *Auxiliary control register (ETH_MACACR)*. When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of *Timestamp status register (ETH_MACTSSR)*.

Flexible pulse-per-second output

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out outputs:

- Fixed pulse-per-second output

  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the *PPS control register (ETH_MACPPSCR)*.
- Flexible pulse-per-second output

  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:

  The start and stop times are programmed through *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.

  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through *PPS width register (ETH_MACPPSWR)* and *PPS interval register (ETH_MACPPSIR)*, respectively.

Note: By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- *the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.*
- *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR) are used only for generating target time reached interrupt; they are not used for PPS output generation.*
- *TRGTMODSEL0/1/2/3 must be programmed to 0.*
- *the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).*

Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the *Subsecond increment register (ETH_MACSSIR)*. The pulse width can be programmed from 1 to 232–1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 232–1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control Register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR) (as shown in Figure 741).

Refer to *Section 61.9.14: Programming guidelines for flexible pulse-per-second (PPS) output* for further details on how configuring flexible pulse output.

PPS start and stop times

The initial start time can be programmed in the Target Time registers.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of *PPS target time nanoseconds register (ETH_MACPPSTTNR)* indicates that the synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS width and interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.

PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through *PTP Offload control register (ETH_MACPOCR)*. 80-bit PTP node identity is configured through the following three registers: *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. *Table 496* indicates the PTP message generation criteria.

Table 496. PTP message generation criteria

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	0	1	Ordinary or Boundary Slave	SYNC message reception	Delay_Req
00	1	1	Ordinary or Boundary Master	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
01	0	1	Transparent Slave	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req

Table 496. PTP message generation criteria (continued)

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
01	1	1	Transparent Master	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
11	X	X	Peer-to-Peer Transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
All other programming combinations are invalid for PTP Offload feature.					

Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP Offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received SYNC *only when the bit is reset.*

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in *Table 496*.

The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

PTP packet generation

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

Table 497. Common PTP message header fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

PTP message header fields

- **messageType**
  The following encoded values are used for PTP message types:
  - SYNC: 0000
  - Delay_Req: 0001
  - Pdelay_Req: 0010
  - Pdelay_Resp: 0011
  - Delay_Resp: 1001
- **transportSpecific**
  The following transport protocol encoding is used:
  - Default PTP over Ethernet: 0000
  - 802.1AS mode: 0001
- **versionPTP**
  It is always set to 2 because PTP version 2 is supported.
- **domainNumber**
  This field contains the value from the *PTP Offload control register (ETH_MACPOCR)*.
- **flagField**
  The following values are used:
  - alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  - twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  - unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address
- **correctionField**
  For more information, see *Table 498*.
- **sourcePortIdentity**
  This field takes the value programmed in the *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.
- **sequenceId**
  Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.
- **controlField**
  The following encoded values are used for controlField:
  - SYNC: 0000 0000
  - Delay_Req: 0000 0001
  - Pdelay_Req: 0000 0010
  - Pdelay_Resp: 0000 0101
  - Delay_Resp: 0000 0011
- **logMessageInterval**
  - SYNC:
    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.

– Delay_Resp:

  This field contains the sum of DRSYNCR and logSyncInterval value taken from the *Log message interval register (ETH_MACLMIR)* for a multicast PTP message and 0111 1111 for unicast PTP message.

– Delay_Req, Pdelay_Req and Pdelay_Resp: 0111 1111

  where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- $2^{(30+N)}$ ns, when N is negative (–1 to –15)
- 2^{N} seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 2^{1}; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is 2^{-1} = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2^{-30} = 1 ns.
- When logSyncInterval is programmed to —-5, the interval is 2^{-5} = 33.55 ms; therefore, the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation $2^{(30+N)}$, because of the non-binary nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:

- **messageLength**

  There is no suffix supported, so this field contains the length of the PTP message that includes 34-byte PTP common header and the body specific to the message type.

  For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.

- **originTimestamp**

  This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req PTP messages.

- **receiveTimestamp**

  For Delay_Resp PTP message, this is the ingress timestamp of the corresponding received Delay_Req PTP message.

- **requestingPortIdentity**

  For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.

- **requestReceiptTimestamp**

  For the Pdelay_Resp PTP message, this field is set to 0.

One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 498 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Timestamp control Register (ETH_MACTSCR)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 498. MAC Transmit PTP mode and one-step timestamping operation

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYP SEL	TSMSTR ENA	TSEVNT ENA		TTSE (2)	OSTC (3)	TTS[4]	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
							Delay_Req (correction field for residence time and egress asymmetric correction)
00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
01	X	0	End-to-end Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Pdelay_Req (correction field for residence time and egress asymmetric correction)
						Ingress TS	Pdelay_Resp (correction field for residence time and ingress asymmetric correction)

Table 498. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[(1)]			Messages processed on transmission
SNAPTYP SEL	TSMSTR ENA	TSEVNT ENA		TTSE (2)	OSTC (3)	TTS[(4)]	
01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay _Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay _Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Delay_Req (correction field for residence time and egress asymmetric correction)

Table 498. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[(1)]			Messages processed on transmission
SNAPTYP SEL	TSMSTR ENA	TSEVNT ENA		TTSE (2)	OSTC (3)	TTS(4)	
11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay _Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

1. The per-packet control values provided here are the recommended settings used by devices in typical PTP operation and for the programmed mode.
2. TTSE represents TTSE bit of TDES2 transmit normal descriptor. The TTSE function is independent from the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of TDES3 transmit context descriptor
4. TTS represents the timestamp value provided in the TTSH, TTSL fields of TDES0 and TDES1 transmit normal descriptor (write-back format).

Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

Enabling one-step timestamp

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

61.5.5 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the Receive path.

Transmit checksum offload engine

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

$$\text{Packet size} < \text{TxQSize} - (\text{PBL} + 5) \times 9$$

where

TxQSize corresponds to the TQS bitfield of *Tx queue x operating mode register (ETH_MTLTXQxOMR)*

PBL corresponds to the TxPBL bitfield of *Channel x transmit control register (ETH_DMACxTXCR)*

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

Note: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in *Table 520: TDES3 normal descriptor (write-back format)*).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in *Table 520: TDES3 normal descriptor (write-back format)*). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 499 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Section : Transmit checksum offload engine because it might result in incorrect checksum insertion or unexpected behavior.

Table 499. Transmit checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Authentication – Any other next header field in main or extension headers	– Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	– Yes – No – Yes – No – No – Yes – No – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	– Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	– No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	– Not applicable – Not applicable	– No – No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive checksum offload engine

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The Receive Checksum Offload Engine (Rx COE) can be enabled by setting the IPC bit of *Operating mode configuration register (ETH_MACCR)*. When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the *VLAN tag register (ETH_MACVTR)* is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the *Rx queue x operating mode register (ETH_MTLRXQxOMR)* is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 500: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 500. Receive checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Any other next header field in main or extension headers	– Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	– Yes – No – Yes – Yes – No – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	– Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	– No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	– Not applicable – Not applicable	– No – No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

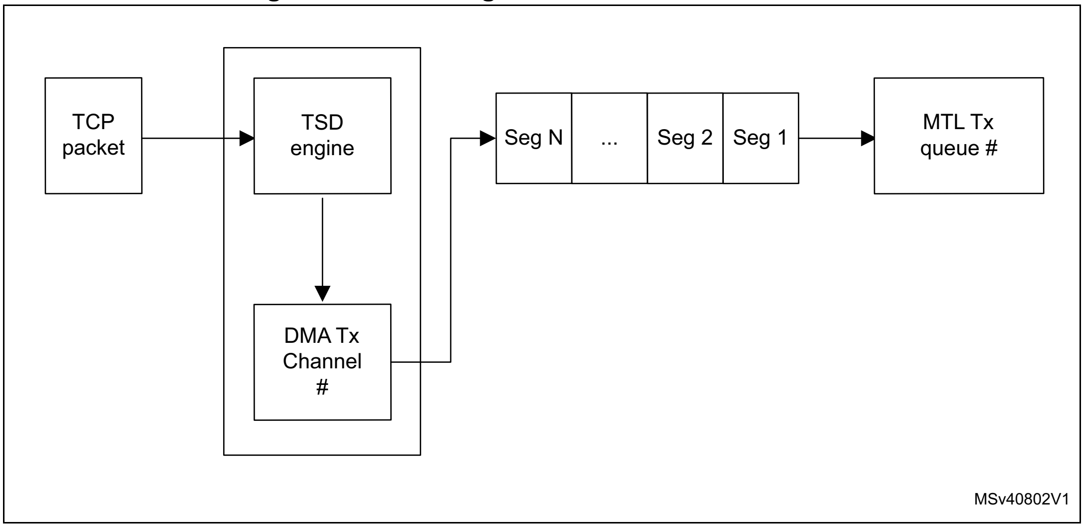
61.5.6 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in *Figure 742*.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACxCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to *Section 61.9.16: Programming guidelines for TSO*.

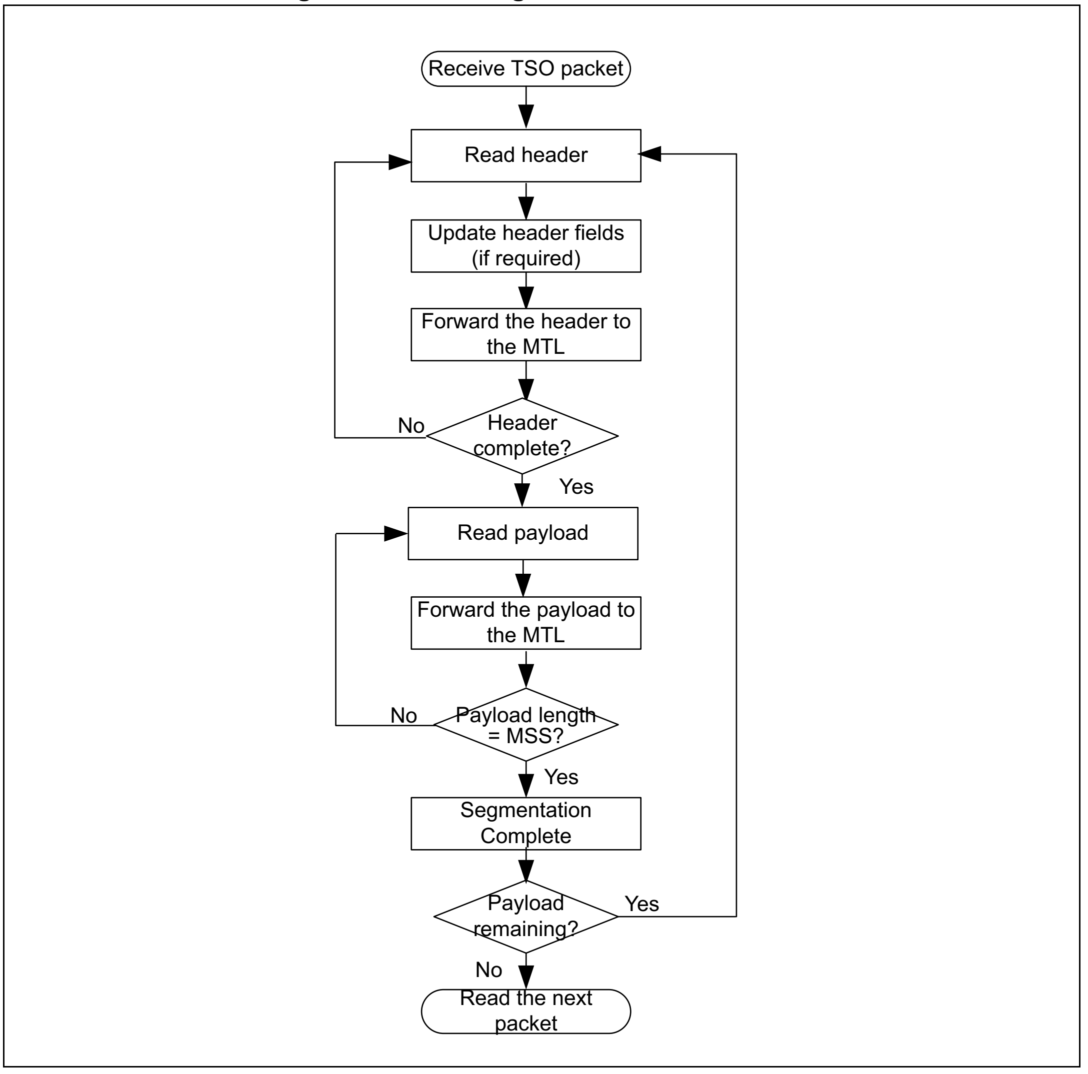
Figure 742. TCP segmentation offload overview



DMA operation with TSO feature

Figure 743 shows the TSO flow.

Figure 743. TCP segmentation offload flow



For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA Tx channel.
3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   – If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.

- If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.
- If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. Table 501 describes how the TCP and IP headers are updated.

Table 501. TSO: TCP and IP header fields

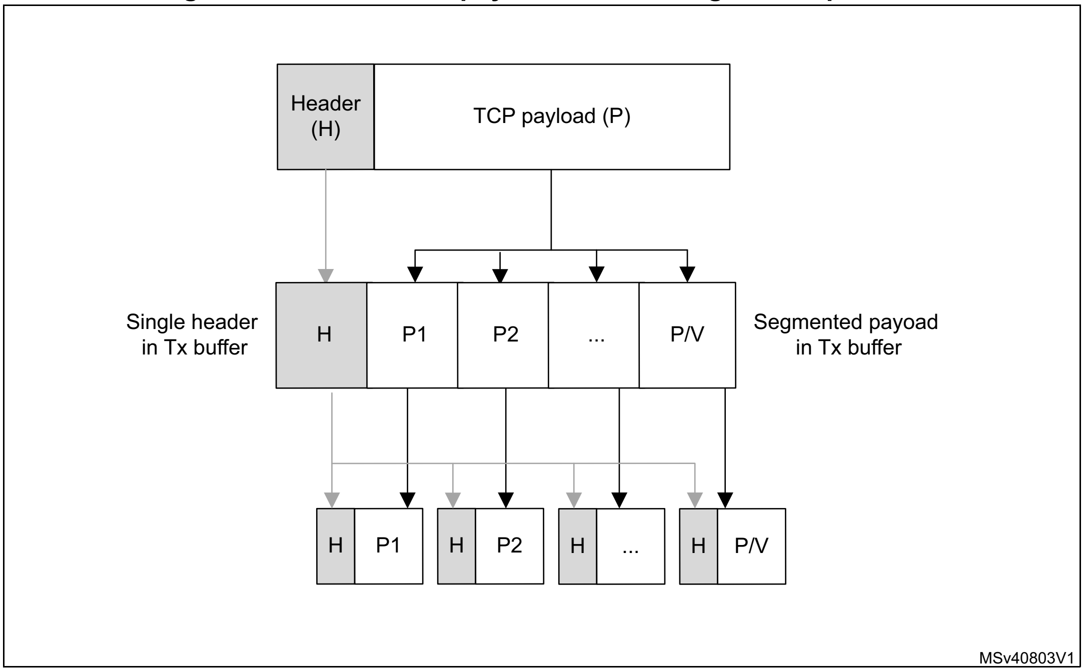
Packet sequence	TCP header	IP header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field is not modified. It is sent as per the header provided by the software. – IPv4 Header Checksum is recalculated. IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field = Previous Identification Field + 1 – IPv4 Header Checksum is recalculated IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = Remaining Payload + TCP Header Length + IP Header Length – Identification Field = Previous Identification Field + 1 – IPv4 header Checksum is recalculated IPv6 Header – Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in *Table 501: TSO: TCP and IP header fields*. *Figure 744: Header and payload fields of segmented packets* shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 744. Header and payload fields of segmented packets



Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to *Section 61.10.3: Transmit descriptor*).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see *Channel x interrupt enable register (ETH_DMACxIER)*).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see *Channel x status register (ETH_DMACxSR)*).

Building the Descriptor and the packet for the TSO feature

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see *Section 61.10.3: Transmit descriptor*). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

Note: The TDES3[22:19] bitfield is used as slot number in case of AV configurations. If both TSO and AV are enabled for the same DMA Tx channel (TSE bit is set in Channel x transmit control register (ETH_DMACxTXCR), then do not enable slot checking for that DMA Tx channel (do not set ESC bit in Channel x slot function control status register (ETH_DMACxSFCSR).

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMACxCR (see *Channel x control register (ETH_DMACxCR)*) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACxTXCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQxOMR register (see *Tx queue x* *operating mode register (ETH_MTLTXQxOMR)*). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

61.5.7 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets**.** ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:

1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier

ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the *Extended operating mode configuration register (ETH_MACECR)* is set, then the MAC does not check for valid CRC of a ARP request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

61.5.8 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using loopback mode

Below some guidelines for using the loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The Transmit and Receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the Receive path. The FIFO is free-running to write on the write clock (eth_tx_clk) and read on every read clock (eth_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of two or four (in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between (G)MII Transmit and Receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling loopback mode

To enable this feature, program the LM bit of the *Operating mode configuration register (ETH_MACCR)*. Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal Receive MII or GMII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on gmii_txd signal.

61.5.9 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and back-pressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

Flow control in Full-duplex mode

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. *Table 502* describes the fields of a Pause packet.

Table 502. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in Half-duplex mode

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 503 describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*
- TFE bit of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.

Table 503. Tx MAC flow control

EHFC	TFE	DM	Description
X	0	X	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set.
1	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set. In addition, the MAC Tx performs backpressure when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*.
0	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set.
1	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set. In addition, the MAC Tx sends a Pause packet when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*.

Transmit flow control

The transmit flow control is enabled when TFE bit is set in *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*.

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding *Tx Queue 0 flow control register (ETH_MACQ0TX-FCR)*.

The flow control is also triggered based on Rx Queue Threshold: the flow control operation of the MAC is enabled when the EHFC bit in the corresponding *Rx queue x operating mode register (ETH_MTLRXQxOMR)* is set. The flow control signal to the MAC is asserted when the fill level of the Rx queue crosses the threshold configured in RFA field of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*. This flow control signal is de-asserted when the fill-level of the queue is lower than the threshold configured in the RFD field.

Note: The RFA sets the threshold at which the flow control is triggered and the MAC schedules to send a PAUSE frame to be sent by the local transmitter.

The remote transmitter can continue to send packets until it receives this PAUSE packet. As a results, the RXQ needs space to take in this data even after the flow control is triggered, in order to avoid overflow and loss of packets.

The maximum space required is calculated theoretically. It can be 2 max-sized packets + a little more assuming that the read from the RxFIFO is not occurring during this whole time.

Theoretically, this makes RFA = 4 or more, which implies that RxQ must have a minimum size of 4 Kbytes.

Practically, the system works with RFA = 2 (full size – 2 Kbytes, with very little probability of overflow).

The RFA bitfield in Rx queue x operating mode register (ETH_MTLRXQxOMR) is used for activating the flow control. These bits control the threshold (fill-level of Rx queue) at which the flow control is activated.

With an RXQ size of 4 Kbytes, RFA must be set to 2 Kbytes to receive the whole packet.

Receive flow control

In the Receive path, the flow control is functional only in Full-duplex mode. If any Pause packet is received in Half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 504 describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of *Rx flow control register (ETH_MACRXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Table 504. Rx MAC flow control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   – Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   – Unicast destination address: the DA matches the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) and the UP bit of *Rx flow control register (ETH_MACRXFCR)* is set.
     If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.
2. The MAC decodes the following fields of the received packet:
   – Type field: this field is checked for 0x8808.
   – Opcode field: this field is checked for 0x0001 (Pause packet).
   – Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

Enabling receive flow control

Set the RFE bit in the *Rx flow control register (ETH_MACRXFCR)* to enable the Pause flow control.

61.5.10 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in *Section 61.11.4: Ethernet MAC and MMC registers*.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the *Packet filtering control register (ETH_MACPFR)*. The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- Six registers used for collision, error, and good packet counters:
  - *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)*
  - *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)*
  - *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)*
  - *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)*
  - *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)*
  - *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)*
- Four registers to record LPI mode transition:
  - *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)*
  - *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)*
  - *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)*
  - *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)*

Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)
  - GMII_RXER input error
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Table 505*.

Table 505. Size of the maximum receive packet

JE	S2KP	GPSLCE	EDVLP	Untagged frame maximum size in bytes	Single VLAN frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

61.5.11 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

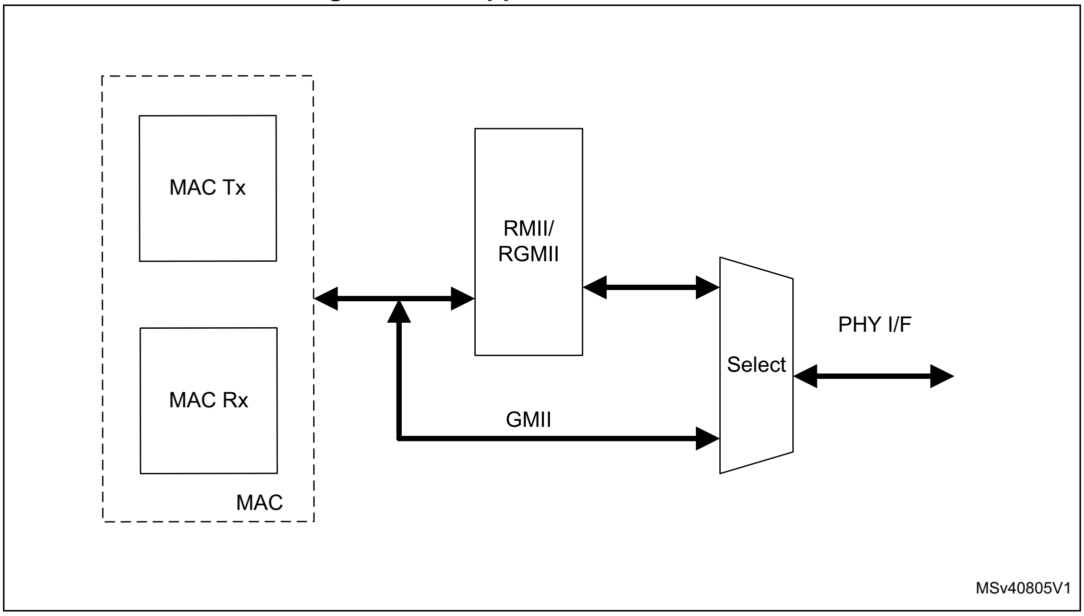
61.5.12 MAC and MMC register descriptions

Refer to *Section 61.11.4: Ethernet MAC and MMC registers*.

61.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII/GMII one. All other interfaces are derived from it as shown in *Figure 745*.

Figure 745. Supported PHY interfaces



This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

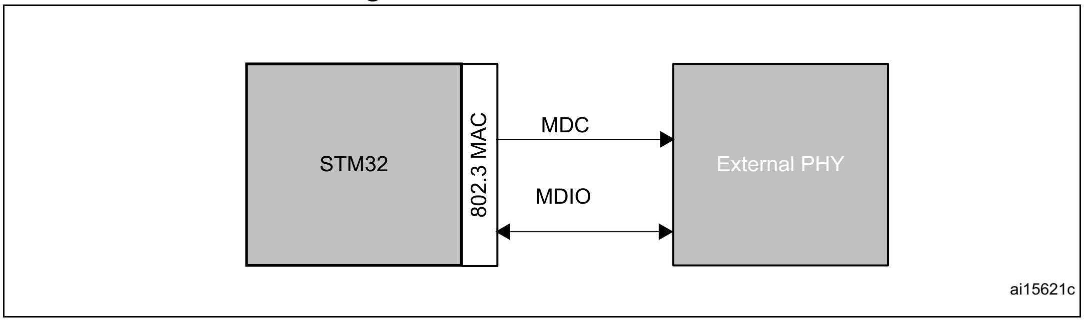
- *Station management agent (SMA)*
- *Media independent interface (MII)*
- *Reduced media independent interface (RMII)*
- *Reduced gigabit media independent interface (RGMII)*

61.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in *Figure 746*.

Figure 746. SMA Interface block



SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the *MDIO address register (ETH_MACMDIOAR)* register. The MDC clock is selected as follows:

Table 506. MCD clock selection

Selection	MDC clock
0000	eth_hclk/42
0001	eth_hclk/62
0010	eth_hclk/16
0011	eth_hclk/26
0100	eth_hclk/102
0101	eth_hclk/124
0110, 0111	-

The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

Figure 747 shows the structure of a Clause 45 MDIO packet, while *Table 507* provides a detailed description of the packet fields.

Figure 747. MDIO packet structure (Clause 45)

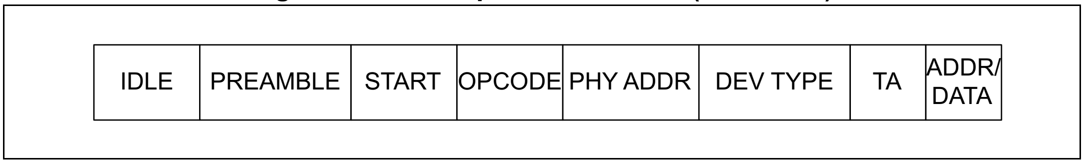


Table 507. MDIO Clause 45 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b00
OPCODE	– 0b00: Address – 0b01: Write – 0b10: Read+ Address – 0b11: Read
PHY ADDR	5-bit address select for one of 32 PHYs
DEV TYPE	5-bit device type
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	16-bit value: for an address cycle (OPCODE = 0b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. – In address and data write cycles, the peripheral drives the ETH_MDIO line during the transfer of these 16 bits. – In read and post-read increment address cycles, the PHY drives the ETH_MDIO line during the transfer of these 16 bits.

The frame structure for Clause 22 frames is also supported. The C45E bit in *MDIO address register (ETH_MACMDIOAR)* can be programmed to enable Clause 22 or Clause 45 mode of operation. *Figure 748* shows the structure of a Clause 22 MDIO packet, while *Table 508* provides a detailed description of the packet fields.

Figure 748. MDIO packet structure (Clause 22)

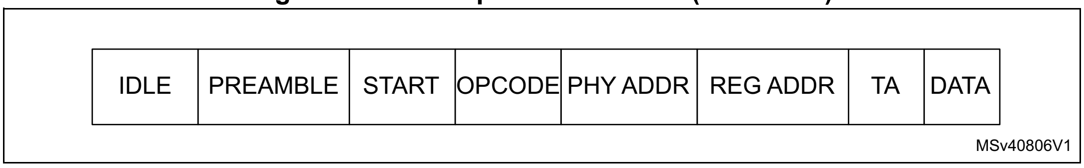


Table 508. MDIO Clause 22 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b01
OPCODE	0b10 for Read and 0b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.

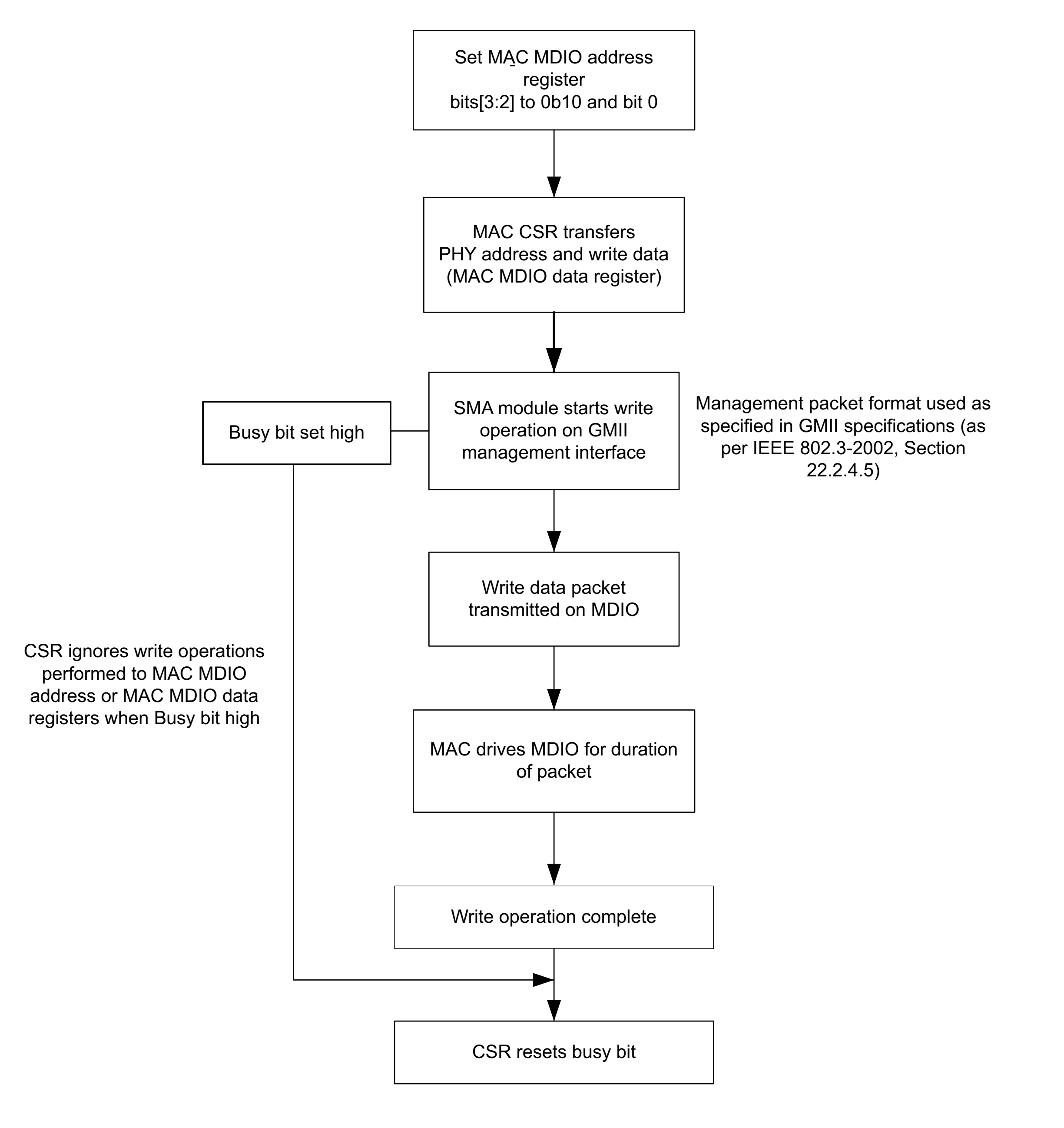
In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

GMII/MII management write operations

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

Figure 749 illustrates the flow for a write operation from the SMA module to the PHY registers.

Figure 749. SMA write operation flow



When bits[3:2] are set to 01 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (*MDIO data register (ETH_MACMDIODR)*) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the GMII management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations performed to the *MDIO address register (ETH_MACMDIOAR)* or the *MDIO data register (ETH_MACMDIODR)* during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 750. Write data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

MSv40807V1

GMII/MII management read operation

When bits[3:2] are set to 11 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the GMII/MII management interface using the management packet format specified in the GMII/MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the *MDIO address register (ETH_MACMDIOAR)* or *MDIO data register (ETH_MACMDIODR)* register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the *MDIO data register (ETH_MACMDIODR)* with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 751. Read data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	10	AAAAA	RRRRR	Z0	DDD...DDD	Z

MSv40808V1

Preamble suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in *MDIO address register (ETH_MACMDIOAR)*.

Trailing clocks and back-to-back transactions

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in *MDIO address register (ETH_MACMDIOAR)* allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in *MDIO address register (ETH_MACMDIOAR)* when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

Interrupt for MDIO transaction completion

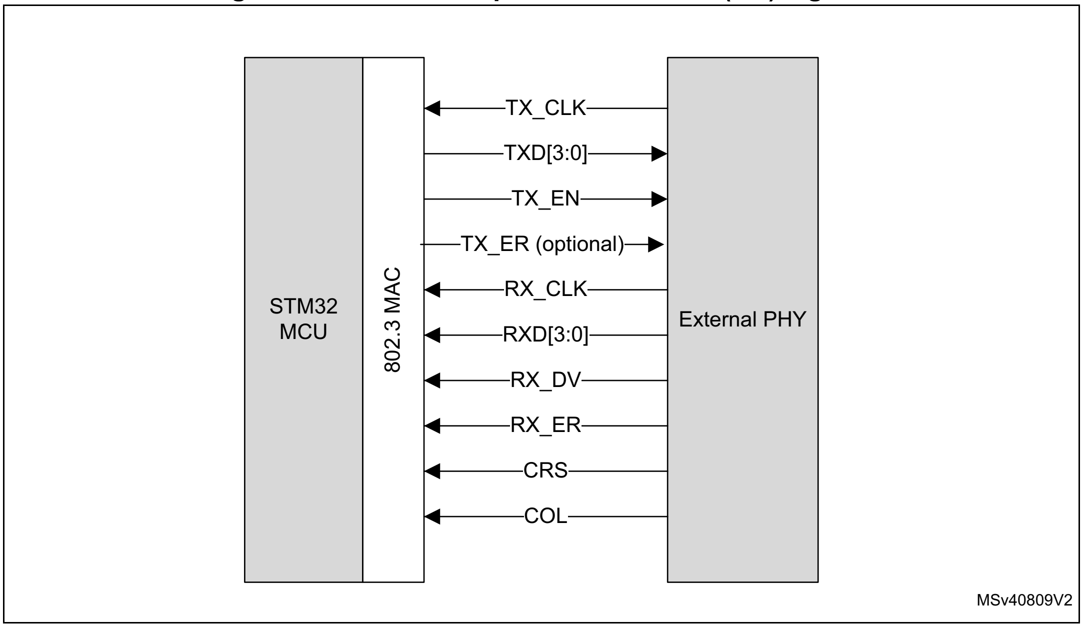
The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the *MDIO address register (ETH_MACMDIOAR)* to know the completion of MDIO commands.

61.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 752: Media independent interface (MII) signals*.

Figure 752. Media independent interface (MII) signals



- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER (optional): required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- RXD[3:0]: receive data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.
- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.
- CRS: carrier sense.

  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don’t care for the MAC sublayer.
- COL: collision detection signal

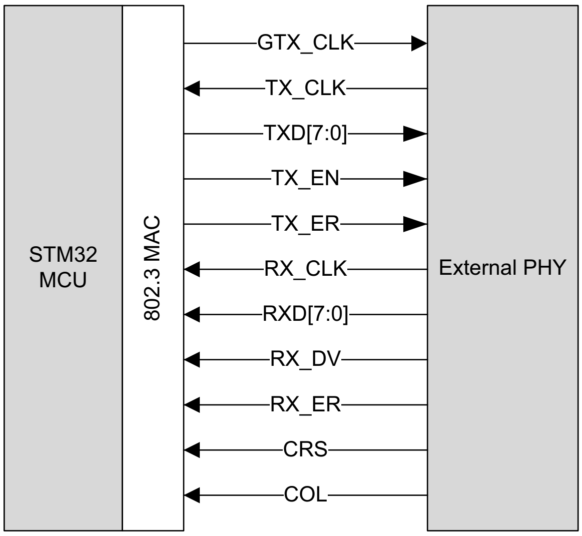
  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don’t care for the MAC sublayer.

61.6.3 Gigabit media independent interface (GMII)

The gigabit media independent interface (GMII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s, 100 Mbit/s and 1000 Mbit/s.

GMII signals are given in *Figure 753*.

Figure 753. Giga media independent interface (GMII) signals



- GTX_CLK: at 1000 Mbit/s, GTX_CLK is output from an STM32 internal oscillator and transmitted to the PHY interface.
- TX_CLK: continuous clock that provides the timing reference for Tx data transfers for 10 Mbit/s and 100 Mbit/s operation. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[7:0]: transmit data

  TXD is a bundle of 8 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[7] is the most significant bit. While TX_EN is deasserted the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER: indicates a Transmit error or carrier extension on TXD[7:0]
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s and 125 MHz at 1000 Mbits/s.
- RXD[7:0]: receive data

  RXD is a bundle of 8 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[7] is the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.
- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.
- CRS: carrier sense.

  This signal is asserted by the PHY when either a transmit or receive medium is non idle. It must be deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don’t care for the MAC sublayer.
- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don’t care for the MAC sublayer.

61.6.4 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU (only in 10/100 mode). According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

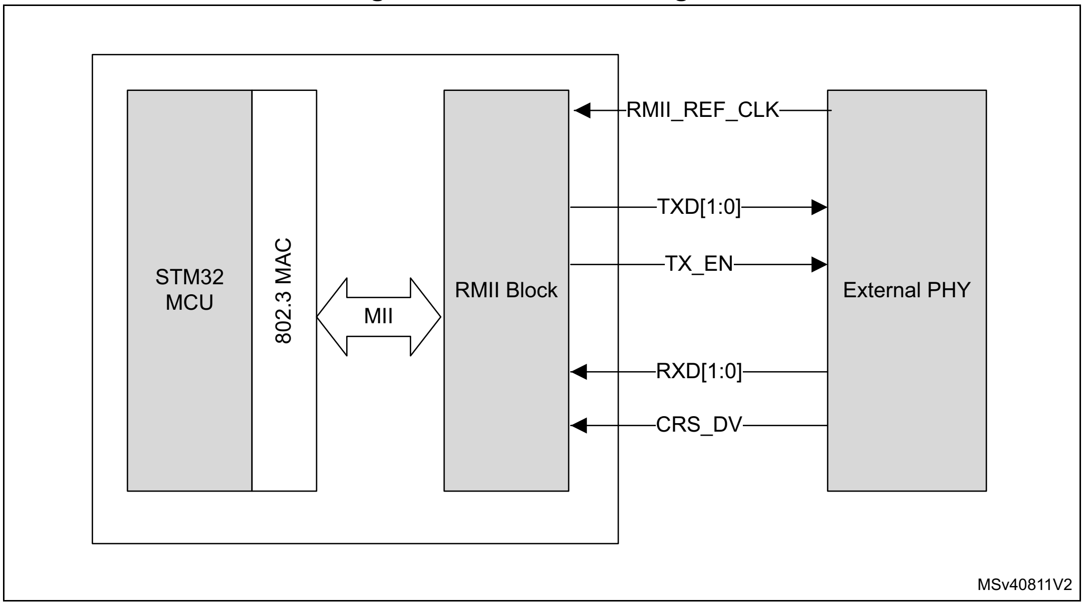
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation.
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.

RMII block diagram

Figure 754: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

Figure 754. RMII block diagram

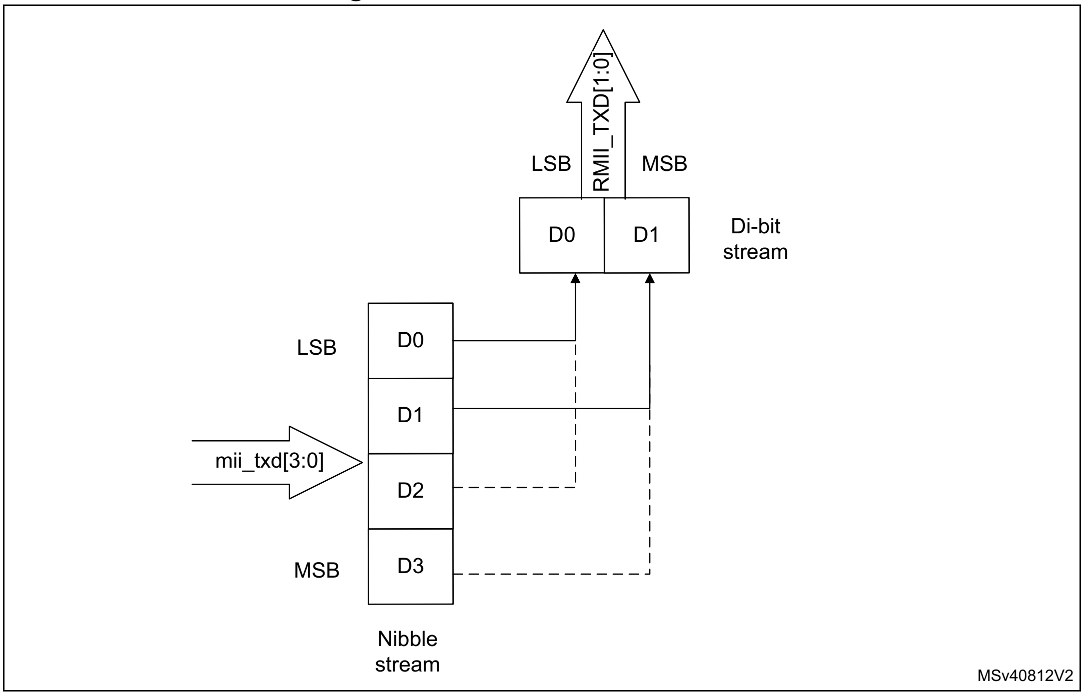


- RMII_REF_CLK: continuous 50 MHz reference clock input
- TXD[1:0]: transmit data
- TX_EN: transmit data enable.
  When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- RXD[1:0]: receive data
- CRS_DV: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.

Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 755: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

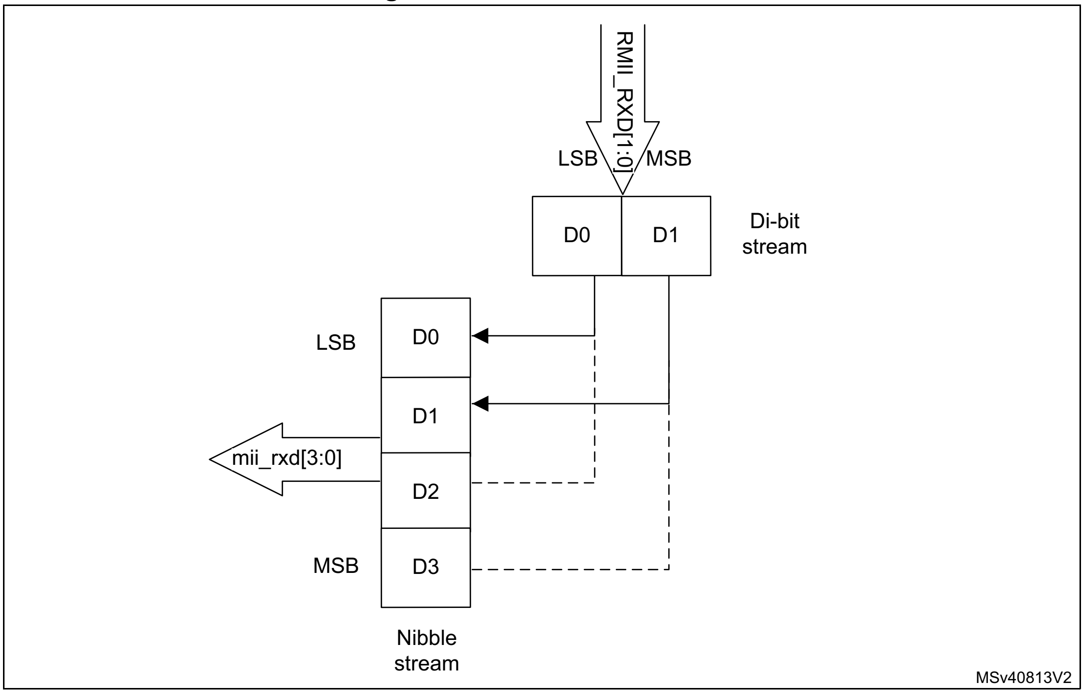
Figure 755. Transmission bit order



Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in *Figure 756: Receive bit order*. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 756. Receive bit order



61.6.5 Reduced gigabit media independent interface (RGMII)

The reduced gigabit media independent interface (RGMII) specification reduces the pin count of the interconnection between the MAC and the PHY for GMII and MII interfaces. To achieve this, the data path and control signals are reduced and multiplexed together with both edges of the Transmit and Receive clocks. For Gigabit operation, the clocks operate at 125 MHz; for 10/100 operation, the clock rates are 2.5/25 MHz.

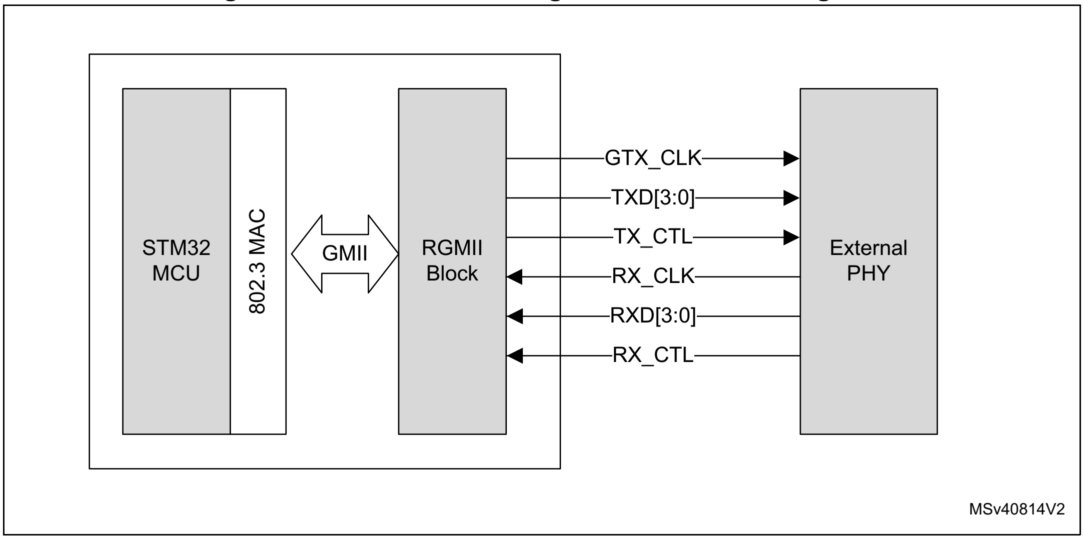
The RGMII module is instantiated between the GMII of the MAC and the PHY to translate the control and data signals between the GMII and RGMII protocols. The RGMII block has the following characteristics:

- Support of 10, 100 and 1000 Mbps operation rates.
- No extra clock required since both edges of the incoming clocks are used.
- Extraction of the in-band (link speed, link status, and duplex mode) status signals from the PHY and provides them to the MAC for link detection.

RGMII block diagram

Figure 757: RGMII block diagram and interface signals shows the position of the RGMII block relative to the MAC and RGMII PHY. The RGMII block is placed between the GMII and the PHY to translate the GMII signals to RGMII signals.

Figure 757. RGMII block diagram and interface signals



- **Transmit path**: this block translates all GMII transmit signals to RGMII signals. The block registers all GMII input signals at the rising edge of eth_gmii_gtx_clk and generates all RGMII transmit signals at the rising and falling edges of eth_gmii_gtx_clk.
- **Receive path:** this block translates all RGMII receive signals to GMII receive signals. The block registers all RGMII input signals at the rising and falling edges of eth_rx_clk and generates all GMII receive signals at the rising edge of eth_rx_clk.
- GTX_CLK: RGMII clock for Tx data transfers, 125 MHz, 25 MHz or 2.5 MHz
- TXD[3:0]: transmit data
- TX_CTL: multiplexing of TX_EN on the rising edge of GTX_CLK, and a logical XOR between TX_EN and TX_ER values on falling edge of GTX_CLK
- RX_CLK: RGMII clock for Rx data transfers
- RXD[3:0]: receive data
- RX_CTL: after demultiplexing, this pin provides ETH_RX_DV on the rising edge of ETH_RX_CLK, and ETH_RX_ER (logical XOR between ETH_RX_CTL values on rising edge and falling edge of ETH_RX_CLK).

Signal conversion

The RGMII block performs data conversions in both directions.

Transmit data conversion

As defined in the RGMII specification, multiplexing of data and control in Gigabit mode is accomplished by using both edges of the Transmit clock. The RGMII block accepts the 8-bit GMII transmit data from the MAC GMII (gmii_txd[7:0]) on the rising edge of eth_gmii_gtx_-clk. For Gigabit operation, the RGMII block then converts gmii_txd[7:0] to two 4-bit nibbles and transmits the data on the RGMII transmit data port (rgmii_txd_o[3:0]) as follows:

- At the rising edge of eth_gmii_gtx_clk, rgmii_txd_o[3:0] contains gmii_txd[3:0]
- At the falling edge of eth_gmii_gtx_clk, rgmii_txd_o[3:0] contains gmii_txd[7:4]

For 10/100 mode, the Transmit data to the PHY is 4 bits. Therefore, for 10/100 mode, the RGMII block drives the gmii_txd[3:0] data to the PHY on rgmii_txd_o[3:0] at the rising edge of eth_gmii_gtx_clk.

Receive data conversion

In 1000-Mbps mode, the RGMII block registers the 4-bit data on the rgmii_rxd_i[3:0] signal at both edges of the Receive clock (eth_rx_clk) and transfers the resulting 8-bit data to the MAC GMII on the rising edge of eth_rx_clk clock, as follows:

- gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of eth_rx_clk
- gmii_rxd[7:4] is the value received on rgmii_rxd_i[3:0] at the falling edge of eth_rx_clk

In 10/100 mode, the RGMII block registers the 4-bit data on rgmii_rxd_i[3:0] only at the rising edge of eth_rx_clk and transfers the data to the MAC GMII on the rising edge of eth_rx-_clk, as follows:

- gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of eth_rx_clk.
- gmii_rxd[7:4] is 0x0.

61.7 Ethernet low-power modes

61.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is in low-power mode, and has to be woken up only at the reception of specific packets from the Ethernet network. In low-power mode, the clocks on most of the host logic, along with the majority of the peripherals (except the MAC receiver logic), can be disabled. On receiving the specific packets from the network, the MAC generates the trigger to wake up the host system and come back to normal state. Refer to the power control (PWR) section of the reference manual for the list of the system operating modes that support Ethernet low-power modes.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and burst traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the magic-packet-based power saving mode is enabled by setting the MGKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.

- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

Magic packet data format

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (*MAC Address 0 high register (ETH_MACA0HR)* and the *MAC Address 0 low register (ETH_MACA0LR)*) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

Destination Address Source Address Length/Type................. FF FF FF FF FF FF

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

...CRC

Description of remote wake-up packet mode

This section describes the power saving mode based on remote wake-up packet.

Note: The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The MAC implements a filter lookup table (programmed through *Remote wake-up packet filter register (ETH_MACRWKPFR)* in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.

- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in *PMT control status register (ETH_MACPCSR)* and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in *Interrupt status register (ETH_MACISR)* is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in *Interrupt enable register (ETH_MACIER)* is set) and CSR clock is not gated off in low-power mode.

Remote wake-up packet filters

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in *Table 509: Remote wake-up packet filter register*.

Table 509. Remote wake-up packet filter register

<table>
<tr><th>ETH_MACRWKPFR value</th><th colspan="8">Field</th></tr>
<tr><td>0</td><td colspan="8">Filter 0 byte mask</td></tr>
<tr><td>1</td><td colspan="8">Filter 1 byte mask</td></tr>
<tr><td>2</td><td colspan="8">Filter 2 byte mask</td></tr>
<tr><td>3</td><td colspan="8">Filter 3 byte mask</td></tr>
<tr><td>4</td><td>Reserved</td><td>Filter 3 command</td><td>Reserved</td><td>Filter 2 command</td><td>Reserved</td><td>Filter 1 command</td><td>Reserved</td><td>Filter 0 command</td></tr>
<tr><td>5</td><td colspan="2">Filter 3 offset</td><td colspan="2">Filter 2 offset</td><td colspan="2">Filter 1 offset</td><td colspan="2">Filter 0 offset</td></tr>
<tr><td>6</td><td colspan="4">Filter 1 CRC - 16</td><td colspan="4">Filter 0 CRC - 16</td></tr>
<tr><td>7</td><td colspan="4">Filter 3 CRC - 16</td><td colspan="4">Filter 2 CRC - 16</td></tr>
</table>

The remote wake-up filter fields are described in *Table 510: Description of the remote wake-up filter fields*.

Table 510. Description of the remote wake-up filter fields

Register	Description
Filter *i* Byte mask	The filter *i* byte mask register defines the bytes of the packet that are examined by filter *i* (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet. – The MSB (31st bit) must be zero. – Bit j[30:0] is the byte mask. – If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter *i* Offset + j of the incoming packet; otherwise Filter *i* Offset + j is ignored.
Filter *i* Command	The 4-bit filter *i* command controls the filter *i* operation. – Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. – Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2". – Bit 1 (And_Previous) implements the Boolean logic(1). When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. – Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.
Filter *i* Offset	This filter *i* offset register defines the offset (within the packet) from which the filter *i* examines the packets. – This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. – The minimum allowed offset is 12, which refers to the 13th byte of the packet. – The offset value 0 refers to the first byte of the packet.
Filter *i* CRC-16	This filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block. – The 16-bit CRC calculation uses the following polynomial: $G(x) = x^{16} + x^{15} + x^2 + 1$ Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following: – 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. – 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

1. The And_Previous bit setting is applicable within a set of four filters.
Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect.
If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered.
If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.

The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through *Remote wake-up packet filter register (ETH_MACRWKPFR)*. The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in *Remote wake-up packet filter register (ETH_MACRWKPFR)* for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of *PMT control status register (ETH_MACPCSR)*.

Note: If the Remote wake-up packet filter register (ETH_MACRWKPFR) is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

When Remote wake-up packet filter register (ETH_MACRWKPFR) is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the Remote wake-up packet filter register (ETH_MACRWKPFR) until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the Remote wake-up packet filter register (ETH_MACRWKPFR) should be at least 4 cycles of the PHY receive clock.

PMT interrupt

The PMT interrupt signal is asserted when a valid remote wake-up packet is received.

Table 511 lists the remote wake-up scenarios in which PMT interrupt is generated.

Table 511. Remote wake-up packet and PMT interrupt generation(1)

Filter i Command			Frame Type and CRC Status		Interrupt Generation
CAST	INV	EN	Received Frame Cast Type	CRC Status	RWK INTR
0	0	1	Unicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
0	1	1	Unicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	0	1	Multicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	1	1	Multicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated

1. In all other combinations, the Remote Wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode.

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the *PMT control status register (ETH_MACPCSR)* is read. This is because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode. When the application clears the PWRDWN bit in *Remote wake-up packet filter register (ETH_MACRWKPFR)*, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the *Channel x transmit control register (ETH_DMACxTXCR)*.
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in *Debug register (ETH_MACDR)* and TXQSTS bit in *Tx queue x debug register (ETH_MTLTXQxDR)* of all MTL Tx Queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in *Operating mode configuration register (ETH_MACCR)*.
- Wait till the Receive DMA empties all frames from the Rx FIFO. You can check this by reading PRXQ[13:0] in *Rx queue x debug register (ETH_MTLRXQxDR)* of all Rx Queues. If these bits are zero, it indicates that the Rx FIFO is empty.
- Configure the magic packet (MGKPKTEN) and/or remote wake-up (RWKPKTEN) detection in the *PMT control status register (ETH_MACPCSR)*.
- Set bit 31 (ARPEN) in the *Operating mode configuration register (ETH_MACCR)*.
- Enable the MAC receiver by setting RE bit and then set PWRDWN bit in the *PMT control status register (ETH_MACPCSR)* to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC transmitter is in the LPI mode when it is put into the power-down mode, then the GMII or MII interface gets clamped to assert the LPI pattern. If the MAC transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  – De-assert the pmt_intr_o by reading the *PMT control status register (ETH_MACPCSR)*.
  – Perform a write operation (with reset values) to the *PMT control status register (ETH_MACPCSR)* and the *Remote wake-up packet filter register (ETH_MACRWKPFR)* so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  – Perform write operations to the *Operating mode configuration register (ETH_MACCR)*, *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)* to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.

After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

61.7.2 Energy Efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps and 1000 Mbps. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of *LPI timers control register (ETH_MACLTCR)*. The PHY Link Status PLS bit of the *LPI control and status register (ETH_MACLCSR)* indicates the link status of the PHY.

Note: The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode. However, the MAC can stop the GTX_CLK clock during the LPI state in the GMII (1000) code.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbps) or TXD[7:0] to 0x01 (for 1,000 Mbps).
4. Updates the status (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:

   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the *LPI timers control register (ETH_MACLTCR)*.
3. Updates the LPI exit status (TLPIEX bit of the *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Figure 758 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 758) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACLCSR) and can be done automatically as shown on Figure 759.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 758. LPI transitions (Transmit, 100 Mbds)

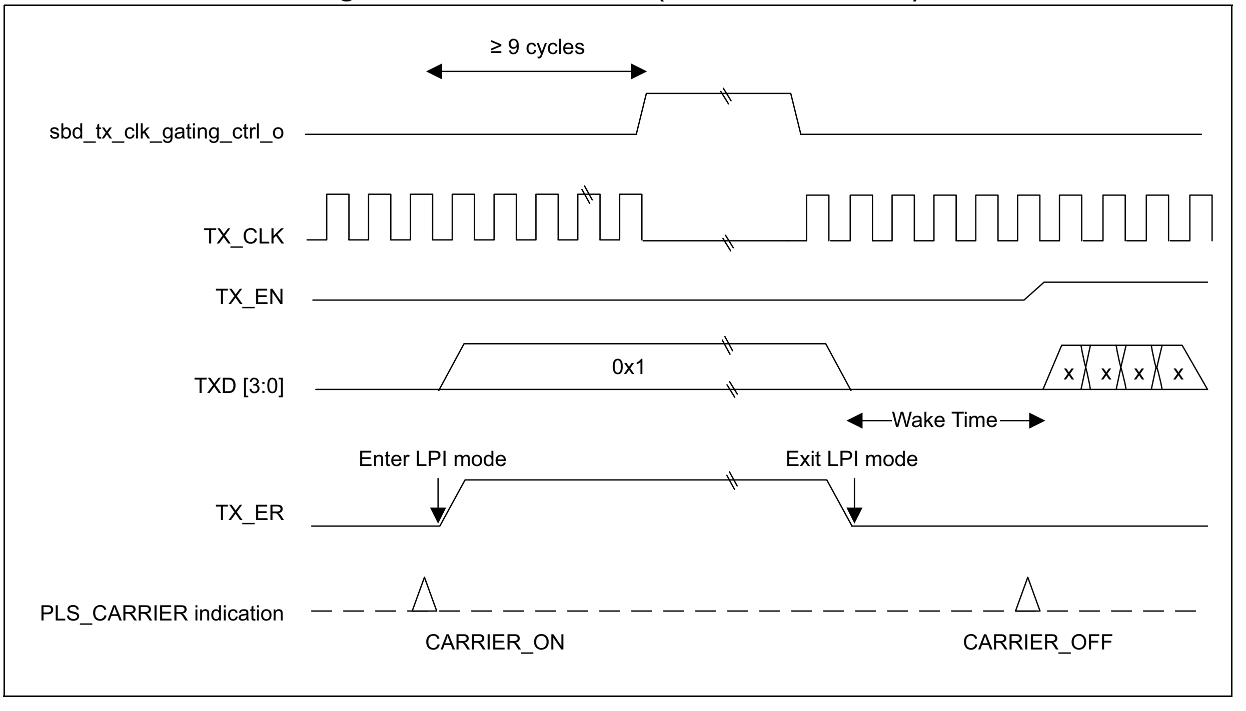
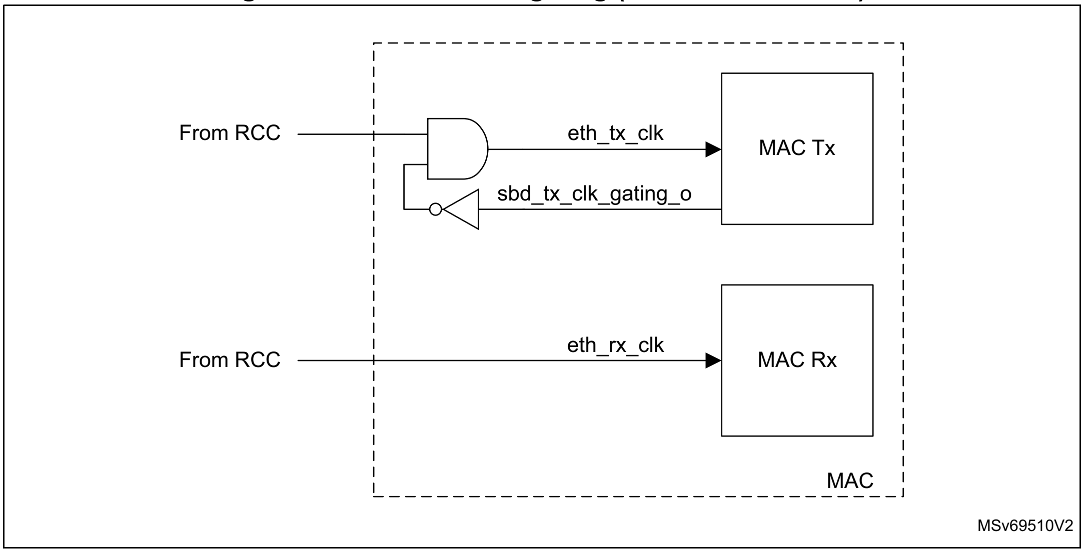


Figure 759. LPI Tx clock gating (when LPITCSE = 1)



Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the *LPI control and status register (ETH_MACLCSR)*.

When LPITXA and LPIEN of *LPI control and status register (ETH_MACLCSR)* are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in *LPI entry timer register (ETH_MACLETR)*. In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbps) or RXD[7:0] to 0x01 (for 1,000 Mbps).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the *LPI control and status register (ETH_MACLCSR)* and immediately generates an interrupt.

Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

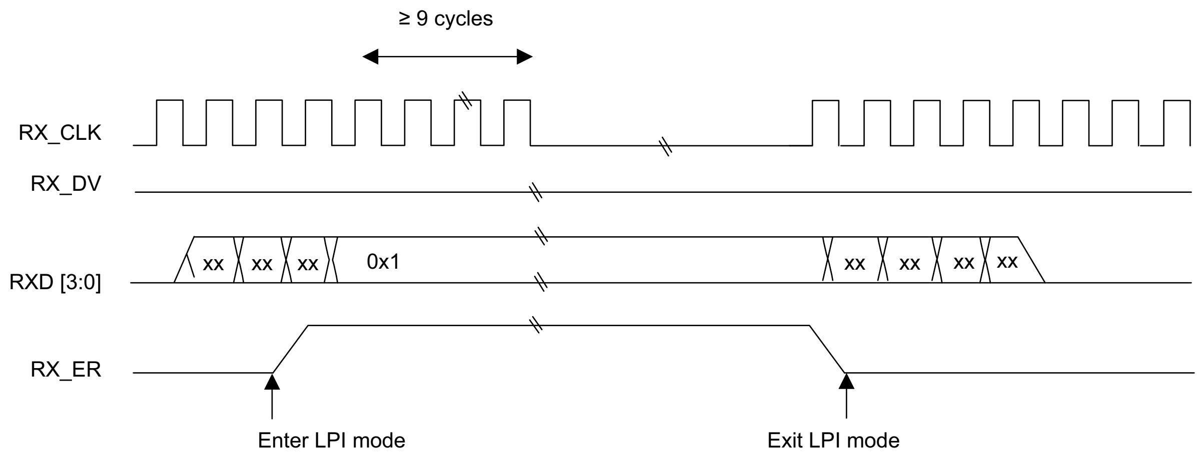
If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the *LPI control and status register (ETH_MACLCSR)* and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 760 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 760. LPI transitions (receive, 100 Mbps)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 760.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the *LPI timers control register (ETH_MACLTCR)* and *LPI entry timer register (ETH_MACLETR)*:

- LPI LS TIMER

  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. You can enable the monitoring of the LUD bit of the *PHYIF control status register (ETH_MACPHYCSR)* by setting the PLSEN bit of *LPI control and status register (ETH_MACLCSR)*.

  The link status is indicated to the MAC by the LNKSTS bit of the *PHYIF control status register (ETH_MACPHYCSR)* or the value programmed by the software in the PLS bit of the *LPI control and status register (ETH_MACLCSR)*. If the link status is not available in the *PHYIF control status register (ETH_MACPHYCSR)*, the software should get the PHY link status by reading the PHY register and accordingly update the PLS bit.

  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the *LPI timers control register (ETH_MACLTCR)*. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.

- LPI TW TIMER

  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of *LPI timers control register (ETH_MACLTCR)*. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.

  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.

- LPI AUTO ENTRY TIMER

  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in *LPI control and status register (ETH_MACLCSR)* is set.

Note: Program the PLS bit of LPI control and status register (ETH_MACLCSR) to 1'b0 before switching between the GMII and MII modes. This resets the internal timers. If the mode is changed after the LPI LS TIMER or LPI TW TIMER starts, the change in the Tx clock frequency can result in incorrect timeout.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the *LPI control and status register (ETH_MACLCSR)*.

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the *LPI control and status register (ETH_MACLCSR)* is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode.

Programming guidelines for Energy Efficient Ethernet

For detailed guidelines on the programming guidelines, see *Section 61.9.13: Programming guidelines for Energy Efficient Ethernet (EEE) on page 3294*.

61.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the *Interrupt status register (ETH_DMAISR)* register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMAC0SR: Channel 0 Status (see *Channel x status register (ETH_DMACxSR)*))
- ETH_DMAC1SR: Channel 1 Status (see *Channel x status register (ETH_DMACxSR)*)
- ETH_MTLISR: Interrupt Status (see *Interrupt status register (ETH_MTLISR)*)
- ETH_MACISR: Interrupt Status (see *Interrupt status register (ETH_MACISR)*)

61.8.1 DMA interrupts

Interrupt registers description

The ETH_DMAC0SR: Channel 0 Status register (see *Channel x status register (ETH_DMACxSR)*) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMAC0IER: Channel 0 Interrupt Enable register (see *Channel x interrupt enable register (ETH_DMACxIER)*) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts.They are indicated by Bits[15:14] of ETH_DMAC0SR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the *Interrupt status register (ETH_DMAISR)* for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of *Interrupt status register (ETH_DMAISR)* are cleared.

Periodic scheduling of Transmit and Receive interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in *Table 515: TDES2 normal descriptor (read format)*) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in*Table 528: RDES3 normal descriptor (read format)*) bit only at some specific intervals of Receive descriptors. This way, whenever

a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer (ETH_DMAC0RXIWTR: Channel 0 Rx Interrupt Watchdog Timer) is given for flexible control and periodic scheduling of Receive interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in *Table 528: RDES3 normal descriptor (read format)*) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMAC0IER register (see *Channel x interrupt enable register (ETH_DMACxIER)*). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Per channel transfer complete interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (*Channel x status register (ETH_DMACxSR)*). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30])).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (*Channel x interrupt enable register (ETH_DMACxIER)*).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (*DMA mode register (ETH_DMAMR)*). *Table 512* explains the behavior of the Transfer Complete interrupt.

Table 512. Transfer complete interrupt behavior

Interrupt mode	Behavior of TI/RI and eth_sbd_intr_it
INTM=0	The TI/RI status signals are set whenever the Transfer complete event is detected. The bits get cleared whenever the software driver writes 1 to these bits. The eth_sbd_intr_it is asserted whenever the corresponding interrupts are also enabled in ETH_DMACxIER register.
INTM=1	The TI/RI is set as explained above. However, the eth_sbd_intr_it is not asserted for any RI/TI events.
INTM=2	The RI/TI status bits are set whenever the Transfer Complete event is detected and gets reset whenever software driver clears those bits by writing 1. However, if another Transfer Complete event is detected before it is cleared (serviced) by the software, these status bits is automatically set again. However, the eth_sbd_intr_it is not generated based on TI/RI.

61.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status register (ETH_MTLISR)* report the queue number responsible for the event. ETH_MTLQ0ICSR: Queue 0 Interrupt Control Status or ETH_MTLQ1ICSR: Queue 1 Interrupt Control Status must be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events on Queue 0 and Queue 1:

- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

61.8.3 MAC Interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:

- Receive Status Interrupt
- Transmit Status Interrupt
- Timestamp Interrupt Status
- MMC Interrupt Status
  – MMC Receive Checksum Offload Interrupt Status
  – MMC transmit interrupt status
  – MMC Receive interrupt status
- LPI Interrupt Status
- PMT Interrupt Status
- PHY Interrupt
- RGMII Interrupt Status

Note: Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.

61.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see *Section 61.9.1*)
- MTL initialization (see *Section 61.9.2*)
- MAC initialization (see *Section 61.9.3*)
- Performing normal receive and transmit operation (see *Section 61.9.4*)
- Stopping and starting transmission (see *Section 61.9.5*)
- Programming guidelines for multichannel multiqueue operation (see *Section 61.9.8*)
- Programming guidelines for GMII link state transitions (see *Section 61.9.9*)
- Programming guidelines for IEEE 1588 timestamping (see *Section 61.9.10*)
- Programming guidelines for AV feature (see *Section 61.9.12*)
- Programming guidelines for Energy Efficient Ethernet (see *Section 61.9.13*)
- Programming guidelines for flexible pulse-per-second (PPS) output (see *Section 61.9.14*)
- Programming guidelines for IEEE 1588 auxiliary snapshot (see *Section 61.9.15*)
- Programming guidelines for TSO (see *Section 61.9.16*)
- Programming guidelines for VLAN filtering on the receiver (see *Section 61.9.17*)

61.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of *DMA mode register (ETH_DMAMR)*).
2. Wait for the completion of the reset process (poll bit 0 of the *DMA mode register (ETH_DMAMR)*, which is cleared when the reset operation is completed).
3. Program the following fields to initialize the *System bus mode register (ETH_DMASBMR)*:
   a) AAL, ONEKBBE, RD_OSR_LMT, and WR_OSR_LMT
   b) Fixed burst or undefined burst
   c) Outstanding requests limit value (OSR_LMT).
   d) If fixed length value is enabled, select the maximum burst length possible on the AXI Bus (bits [7:1])
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to *Section 61.10: Descriptors*.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMAC0TXRLR and ETH_DMAC0RXRLR registers (see *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (*Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*, *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*). In addition, program the transmit and receive tail pointer

registers that inform the DMA about the available descriptors (see *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)*).

7. Program ETH_DMAC0CR, ETH_DMAC0TXCR and ETH_DMAC0RXCR registers (see *Channel x control register (ETH_DMACxCR)*, *Channel x transmit control register (ETH_DMACxTXCR)* and *Channel 0 receive control register (ETH_DMAC0RXCR)*) to configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.
8. Enable the interrupts by programming the ETH_DMAC0IER register (see *Channel x interrupt enable register (ETH_DMACxIER)*).
9. Start the Receive and Transmit DMAs by setting SR (bit 0) of *Channel 0 receive control register (ETH_DMAC0RXCR)* and ST (bit 0) of the ETH_DMAC0TXCR (see *Channel x transmit control register (ETH_DMACxTXCR)*).
10. Repeat steps 4 to 9 for all the Tx DMA channels selected in the hardware.

61.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the Tx Scheduling (SCHALG) and Receive Arbitration Algorithm (RAA) fields in *Operating mode register (ETH_MTLOMR)* to initialize the MTL operation when multiple Tx and Rx queues are used.
2. Program the following fields to initialize the operating mode in *Tx queue x operating mode register (ETH_MTLTXQxOMR)*.
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).
3. Program the following fields to initialize the operating mode in the ETH_MTLRXQxOMR register (see *Rx queue x operating mode register (ETH_MTLRXQxOMR)*):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).
4. Repeat the previous two steps for all the MTL Tx and Rx queues selected in the configuration

61.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: *MAC Address x low register (ETH_MACAxLR)*, *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x high register (ETH_MACAxHR)*.
2. Program the following fields to set the appropriate filters for the incoming frames in the *Packet filtering control register (ETH_MACPFR)*:
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*:
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.
   c) Flow Control Busy.
4. Program the *Interrupt enable register (ETH_MACIER)* as required, if it is applicable for your configuration.
5. Program the appropriate fields in the *Operating mode configuration register (ETH_MACCR)* register. For example, Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in *Operating mode configuration register (ETH_MACCR)* register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JE bit to 1.
  b) Set JD and WD bits to 0 to avoid giant packet error reporting.
  c) Set GPSLCE bit to 1
  d) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JD and WD bits to 1 to avoid giant packet error reporting.
  b) Set GPSLCE bit to 1.
  c) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to 16383.

61.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the

descriptors and writing the ETH_DMAC0TXDTPR (see *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and ETH_DMAC0RXDTPR (see *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)*).

4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMAC0CATXDR and ETH_DMAC0CARXDR registers (see *Channel x current application transmit descriptor register (ETH_DMACxCATXDR)* and *Channel 0 current application receive descriptor register (ETH_DMAC0CARXDR)*).
5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMAC0CATXDR and ETH_DMAC0CARXDR registers (see *Channel x current application transmit descriptor register (ETH_DMACxCATXDR)* and *Channel 0 current application receive descriptor register (ETH_DMAC0CARXDR)*).

61.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMAC0TXCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)* (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the *Operating mode configuration register (ETH_MACCR)* Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)*, PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in *Tx queue x debug register (ETH_MTLTXQxDR)* and RXQSTS[1:0] is set to 00.
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active.

Similarly, do not change the DMA-related configuration when Transmit and Receive DMA are active.

61.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an RxFIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

61.9.7 Programming guidelines for switching the AXI clock frequency

To dynamically change the AXI clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the *Tx queue x debug register (ETH_MTLTXQxDR)* and the status of DMA is given in *Debug status register (ETH_DMADSR)*.
2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in *Operating mode configuration register (ETH_MACCR)*.
3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in *Rx queue x debug register (ETH_MTLRXQxDR)*.
4. Ensure that the application does not perform any register read or write operation.
5. Change the frequency of the AXI clock.
6. Enable the MAC transmitter or the MAC receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

61.9.8 Programming guidelines for multichannel multiqueue operation

Transmit path

1. Program the Transmit queue size in the TQS field in ETH_MTLTXQ**x**OMR register. The size of the queue is determined by the value programmed in the TQS field.

   During the Transmit operation, the number of channels is equal to the number of the queues. As a result, the channel to queue mapping is fixed.
2. To use a queue, the queue must be enabled by setting the TXQEN bit in the corresponding ETH_MTLTXQ**x**OMR register. The ST bit in *Channel x transmit control register (ETH_DMACxTXCR)* and the corresponding TXQEN bit in ETH_MTLTXQ**x**OMR register must be enabled.
3. Program the scheduling method in SCHALG bit of ETH_MTLOMR register.
4. If CBS algorithm is enabled on AV queue 1, ETH_MTLTXQ1ECR, ETH_MTLTXQ1SSCR, ETH_MTLTXQ1HCR and ETH_MTLTXQ1LCR registers also need to be programmed as required.

Receive path

1. Program the Receive queue size in the RQS field in ETH_MTLRXQ**x**OMR register. The size of the queue is determined by the value programmed in the RQS field.
2. Enable the Receive queues 0 to 1 in the RXQ0EN to RXQ1EN fields of ETH_MACRXQ0CR register for AV.

   Enable SR bit of statically or dynamically mapped *Channel 0 receive control register (ETH_DMAC0RXCR)* and corresponding RXQiEN in *Rx queue control 0 register (ETH_MACRXQC0R)*.
3. The MAC routes the Rx packets to the Rx queues depending on following packet types:
   a) AV PTP packets are routed according to the value of PTPQ in ETH_MACRXQC1R register

b) AV untagged control packets are routed according to the value of AVCPQ in ETH_MACRXQC1R register.

c) VLAN tag priority field in VLAN tagged packets

Program PSRQ1 and PSRQ0 of the ETH_MACRXQC2R register to configure the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 0 to 1.

d) The AV tagged control and data packets are also routed according to PSRQ field of the ETH_MACRXQC2R register.

Note: The priorities set in PSRQ0 and PSRQ1 must be unique.

Programming guidelines for recovering from DMA channel failure

When the DMA channel issues a bus error, follow the steps described below to recover from the failure.

Recovering from the receive DMA channel failure

Follow these steps if a bus error occurred in the Receive DMA channel:

1. Set the RPF bit to 1. This flushes all the packets one after the other. This step is optional. However, setting this bit prevents HOL (head-of-line) blocking in the Rx queues when packets sent to the RXDMA are stopped due to bus error.
2. Reprogram the specific registers of the DMA channel.
3. Start the DMA channel.

Recovering from the transmit DMA channel failure

1. Stop the specific DMA channel, even if it is in active state.
2. Flush the corresponding MTL queue.
3. Reprogram the specific registers of the DMA channel.
4. Start the DMA channel.

Note: Due to the known limitations, reprogramming the DMA channel registers might not always be successful to recover from a bus error. If the peripheral is not fully functional after reprogramming the DMA, as a workaround, issue a soft reset to recover from the bus error.

Programming guidelines for disabling receive queue

Follow these steps to disable specific receive queue:

1. Disable the receiver by setting the RE bit of the *Operating mode configuration register (ETH_MACCR)* to 0.
2. Read the *Debug register (ETH_MACDR)* (ensure that RPESTS and RFCFCSTS fields are 0) and *Rx queue x debug register (ETH_MTLRXQxDR)* of all the Rx queues.
3. When all *Rx queue x debug register (ETH_MTLRXQxDR)* are read as 0, disable the intended Rx queue by programming the corresponding fields in the *Rx queue control 0 register (ETH_MACRXQC0R)*.
4. Enable the receiver by setting the RE bit of *Operating mode configuration register (ETH_MACCR)* to 1.

61.9.9 Programming guidelines for GMII link state transitions

Transmit and Receive clocks are running when the link is down

Complete the following steps when the link is down while the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of *Channel x control register (ETH_DMACxCR)*.
2. Disable the MAC receiver by clearing RE bit of *Operating mode configuration register (ETH_MACCR)*.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)* (TRCSTS[1:0] is not 01).

   or

   Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing TE bit of the *Operating mode configuration register (ETH_MACCR)* Register.
5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in *Tx queue x debug register (ETH_MTLTXQxDR)* and RXQSTS[1:0] to 00 in *Rx queue x debug register (ETH_MTLRXQxDR)*).
6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.
7. Restart the operation by starting the Tx DMA. Then enable the MAC transmitter and receiver.

   The Rx DMA does not need to be enabled: since the receiver is disabled, there are no data in the Rx FIFO.

Transmit and Receive clocks are stopped when the link is down

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped:

1. Disable the MAC transmitter and receiver by clearing RE and TE bits in the *Operating mode configuration register (ETH_MACCR)*. This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register (ETH_MACDR)* (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

61.9.10 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control Register (ETH_MACTSCR)*. However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of *Interrupt enable register (ETH_MACIER)*.
2. Set bit 0 of *Timestamp control Register (ETH_MACTSCR)* to enable timestamping.
3. Program *Subsecond increment register (ETH_MACSSIR)* based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register (ETH_MACTSAR)* and set bit 5 of *Timestamp control Register (ETH_MACTSCR)*.
5. Poll the *Timestamp control Register (ETH_MACTSCR)* until bit 5 is cleared.
6. Program bit 1 of *Timestamp control Register (ETH_MACTSCR)* to select the Fine Update method (if required).
7. Program *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)* with the appropriate time value.
8. Set bit 2 in *Timestamp control Register (ETH_MACTSCR)*.

   The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:

   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

   b) Program *Timestamp ingress asymmetric correction register (ETH_MACTSIACR)* to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control Register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.

System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (*System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*).
2. Set bit 3 (TSUPDT) of the *Timestamp control Register (ETH_MACTSCR)*.

   The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in *Section : System time register module*, calculate at which rate you intend to increment or decrement the system time.
2. Update the *Timestamp addend register (ETH_MACTSAR)* with the new value and set bit 5 of the *Timestamp control Register (ETH_MACTSCR)* Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.
5. Enable the Timestamp interrupt in bit 12 of *Interrupt enable register (ETH_MACIER)*.
6. Set bit 4 in Register *Timestamp control Register (ETH_MACTSCR)*.
7. When this trigger generates an interrupt, read *Interrupt status register (ETH_MACISR)*.
8. Reprogram *Timestamp addend register (ETH_MACTSAR)* with the old value and set bit 5 again.

61.9.11 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPTYPSEL, TSMSTRENA, and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCEN bit of *PTP Offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Sync message.
5. Program the LSI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Sync messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Sync message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Sync message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable periodic generation of PTP Pdelay_Req messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.
3. Program the APDREQEN bit of *PTP Offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Pdelay_Req message.
5. Program the LMPDRI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Pdelay_Req messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the DRSYNCR and LSI fields in *Log message interval register (ETH_MACLMIR)*. The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the DRSYNCR field in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated)

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
4. Program the DRSYNCR and LMPDRI fields in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Master mode

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the DRSYNCR, LSI and LMPDRI fields in *Log message interval register (ETH_MACLMIR)*, the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):

1. Program the SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 3, X, and X respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP*

Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.

4. Program the LMPDRI field in *Log message interval register (ETH_MACLMIR)* to indicate periodicity of the PTP Pdelay_Req messages
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

61.9.12 Programming guidelines for AV feature

Initializing the DMA

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all registers and logic (bit 0 in ETH_DMAMR).
2. Wait for the completion of the reset process. Poll bit 0 of the ETH_DMAMR, which is cleared only when the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in ETH_DMAMR register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the Transmit and Receive descriptors. When OSF mode is used, at least two TX descriptors are required. For more information about descriptors, refer to *Section 61.10: Descriptors*.
5. Make sure that your software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the Transmit and Receive Ring length registers (*Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*). The ring length programmed must be at least 4.
7. Initialize the receive and transmit descriptor list address with the base address of the transmit and receive descriptor (*Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*, *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*). In addition, program the Transmit and Receive tail pointer registers indicating the available descriptors to the DMA (*Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)*).
8. Program the following fields to initialize the operating mode in the ETH_MTLTXQ0OMR:
   a) Transmit Store And Forward (TSF)
   b) Transmit Threshold Control (TTC)
   c) Transmit Queue Enable (TXQEN) to value 10 to enable Transmit Queue0
   d) Transmit Queue Size (TQS)
9. Enable the interrupts by programming the EH_DMAC0IER.
10. Repeat steps 4 to 9 for all the Tx DMA and RX DMA channels.

11. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues.
12. Start the Receive and Transmit DMA by setting bit 0 of the ETH_DMAC0TXCR and bit 0 of the ETH_DMAC0RXCR (see *Channel x transmit control register (ETH_DMACxTXCR)* and *Channel 0 receive control register (ETH_DMAC0RXCR)*).

Enabling slot number checking

You can use the slot number check feature to specify the intervals at which the DMA Channels mapped to AV queues fetches the frames from the AHB/AXI system bus. This feature is useful to ensure a uniform and periodic transfer of the AV traffic from the host memory. It is available only when timestamping is enabled and the *Subsecond increment register (ETH_MACSSIR)* is programmed. Complete the following steps to enable the slot number checking:

Note: The slot number checking steps should be complete after programming the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues (step 11 of Section : Initializing the DMA). and before starting the Receive and Transmit DMA (step 12).

1. Enable timestamping by following the steps described in *Section : Initializing the System time generation*.
2. Make sure that the SLOTNUM field (bits 22 to 19) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. To do this, read the current reference slot number from the *Channel x slot function control status register (ETH_DMACxSFCSR)*.
3. Set bit 0 (ESC) of *Channel x slot function control status register (ETH_DMACxSFCSR)* to enable the slot number checking.

Enabling average bits per slot reporting

The CBS Status register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. Enable timestamping by following the steps described in *Section : Initializing the System time generation*.
2. Program SLC bits of the *Tx queue 1 ETS control register (ETH_MTLTXQ1ECR)* of a channel with number of slots over which the average transmitted bits per slot need to be computed.
3. Enable ABPSIE of the *Queue x interrupt control status register (ETH_MTLQxICSR)* of a channel to generate the average bits per slot interrupt.

Note: The frequency of this interrupt depends on the value programmed during previous step. For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.

When it is not required, you can disable this interrupt to stop the interrupt flooding.

4. Read ABS bits from *Tx queue x ETS status register (ETH_MTLTXQxESR)* of a channel on each interrupt.

   The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 1 (ABPSIS) of *Queue x interrupt control status register (ETH_MTLQxICSR)* indicates that a new value is updated in the ABS field.

Disabling flow control for AV-enabled queues

Transmit flow

Program the EHFC (Enable Hardware Flow Control) bit of corresponding *Rx queue x operating mode register (ETH_MTLRXQxOMR)* to 0.

61.9.13 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).
3. Program bits 25 to 16 and bits 15 to 0 in *LPI timers control register (ETH_MACLTCR)*.
4. Read the PHY link status by using the MDIO interface and update bit 17 of *LPI control and status register (ETH_MACLCSR)*.

   Update *LPI control and status register (ETH_MACLCSR)* accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program *One-microsecond-tick counter register (ETH_MAC1USTCR)* as per the frequency of the clock used for accessing the CSR slave port.
6. Program the LPIET bit in the *LPI entry timer register (ETH_MACLETR)* with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bits 20 to 19) of *LPI control and status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
8. Set bit 16 of *LPI control and status register (ETH_MACLCSR)* to put the MAC transmitter in LPI state.

   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.
9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
10. The MAC transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.
11. Reset LPIEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC transmitter from LPI state.

Note: To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in LPI control and status register (ETH_MACLCSR).

To switch off the GMII Transmit clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal to gate the clock input.

To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of LPI control and status register (ETH_MACLCSR) to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

Gating Off the CSR Clock in the LPI mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

Gating off the CSR clock in the Rx LPI mode

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when *LPI control and status register (ETH_MACLCSR)* is read.

Gating off the CSR clock in the Tx LPI mode

The following operations are performed when bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off, the events on the MAC receiver do not get reported or updated in the CSR. To restore the CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR clock is resumed, reset bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* to exit the MAC from LPI mode.

61.9.14 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS*

target time seconds register (ETH_MACPPSTTSR) and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) as start time of PPS signal output.

2. Program the start time value in the Target Time registers (register *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
3. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)* Register.
4. Program PPSCMD[3:0] of *PPS control register (ETH_MACPPSCR)* to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in *PPS width register (ETH_MACPPSWR)*.
3. Program PPSCMD[3:0] bits of *PPS control register (ETH_MACPPSCR)* to generate a single pulse after the previous pulse is deasserted. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in Target Time registers.

   If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the peripheral may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS*

target time seconds register (ETH_MACPPSTTSR) and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) for start time of the PPS signal output.

2. Program the start time value in the Target Time registers (register *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
3. Program the interval value between the train of pulses on the PPS signal output in *PPS interval register (ETH_MACPPSIR)*.
4. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)*.
5. Program PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands.
6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in *PPS target time nanoseconds register (ETH_MACPPSTTNR)* is reset before programming the Target Time registers again.
7. Program the PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field.

Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed.

The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL[1:0] bits in *PPS control register (ETH_MACPPSCR)* enable you to program the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of *PPS control register (ETH_MACPPSCR)* to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

Note: The TSTRGTERR0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to *the system time in the System time seconds register (ETH_MACSTSR) and System time nanoseconds register (ETH_MACSTNR).*

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register (ETH_MACIER) is set.

Therefore, to avoid unwanted interrupt, the correct writing order is as follow:

1. *PPS target time nanoseconds register (ETH_MACPPSTTNR).*
2. *PPS target time seconds register (ETH_MACPPSTTSR).*
3. *PPS interval register (ETH_MACPPSIR).*
4. *PPS width register (ETH_MACPPSWR).*
5. *PPSCTRL[3:0] and PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register (ETH_MACPPSCR).*

61.9.15 Programming guidelines for IEEE 1588 auxiliary snapshot

The IEEE 1588 auxiliary snapshot function is used to capture the auxiliary snapshot for a trigger event received on the ptp_aux_ts_trig_i input.

To program the IEEE 1588 auxiliary snapshot function, complete the following steps:

1. Program the TSIE bit in the *Interrupt enable register (ETH_MACIER)* to enable interrupt generation when the auxiliary snapshot is captured in the FIFO.
2. Program the ATSENx bit of the *Auxiliary control register (ETH_MACACR)* to enable the capture of the auxiliary snapshot based on the respective trigger input.
3. If the software does not want the captured auxiliary snapshot to be read, program the ATSFC bit of *Auxiliary control register (ETH_MACACR)* anytime to clear the auxiliary snapshot FIFO.
4. When TSIS is set or an interrupt is generated (due to TSIE programmed to 1), read the *Timestamp status register (ETH_MACTSSR)*, *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)* to access the captured auxiliary snapshot.

Note: When the auxiliary snapshot is captured in the FIFO for a trigger event, the following bitfields are updated:

- *AUXTSTRIG (status bit indicating that an auxiliary snapshot has been captured), ATSSTN (trigger event input source identifying the first available entry in the FIFO), and ATSNS (indicating the number of auxiliary snapshots available in the FIFO) bitfields of the Timestamp status register (ETH_MACTSSR).*
- *TSIS (indicating that an interesting event occurred in the IEEE 1588 timestamp function) bitfield in the Interrupt status register (ETH_DMAISR).*

The AUXTSTRIG bit is cleared on read (or set when the RCWE bit of the CSR software control register (ETH_MACCSRSWCR) is set).

Note: When the captured auxiliary snapshot is ignored because that the FIFO is full, the ATSSTM bit, which indicates that a trigger has been missed, is set.

61.9.16 Programming guidelines for TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding *Channel x transmit control register (ETH_DMACxTXCR)* to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
   – MSS[13:0] of *Channel x control register (ETH_DMACxCR)*
   – or MSS in the context descriptor

     If MSS[13:0] field is programmed in both *Channel x control register (ETH_DMACxCR)* and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.

61.9.17 Programming guidelines to perform VLAN filtering on the receiver

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program *VLAN tag register (ETH_MACVTR)* for the following bit to select the filtering method:
   – ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   – VTHM: VLAN Tag Hash Table Match Enable.
   – ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   – ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   – DOVLTC: Ignores VLAN Type for Tag Match
   – VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL bit in *VLAN tag register (ETH_MACVTR)* for the 12-bit or 16-bit VLAN tag.
3. If VLAN tag Hash filtering is enabled, program *VLAN Hash table register (ETH_MACVHTR)*.
   – When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the *VLAN Hash table register (ETH_MACVHTR)*.
   – When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of *VLAN Hash table register (ETH_MACVHTR)*.

   For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

61.10 Descriptors

61.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**
  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- **Context descriptors**
  The context descriptors are used to provide control information applicable to the packet to be transmitted.

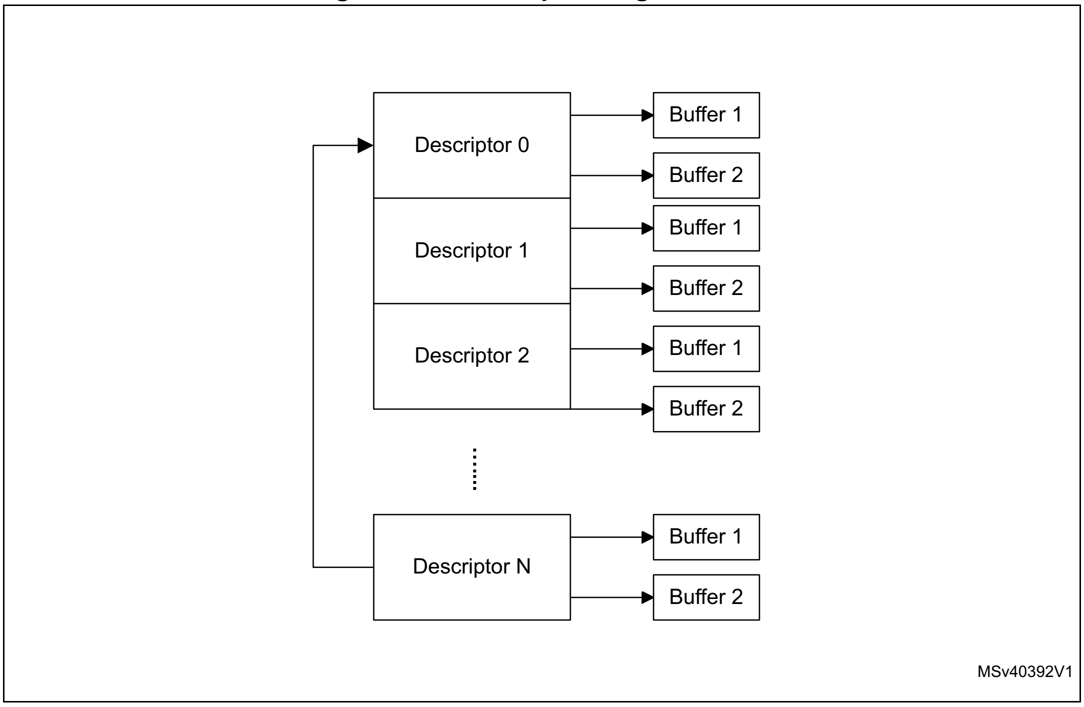
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.

61.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 761. Descriptor ring structure



In a ring structure, descriptors are separated by the 64-bit word number programmed in the DSL field of the *Channel x control register (ETH_DMACxCR)*). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)*)
- *Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)*)

The *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* or *Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)* contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the descriptor tail pointer register and update the tail pointer so that the following condition is met:

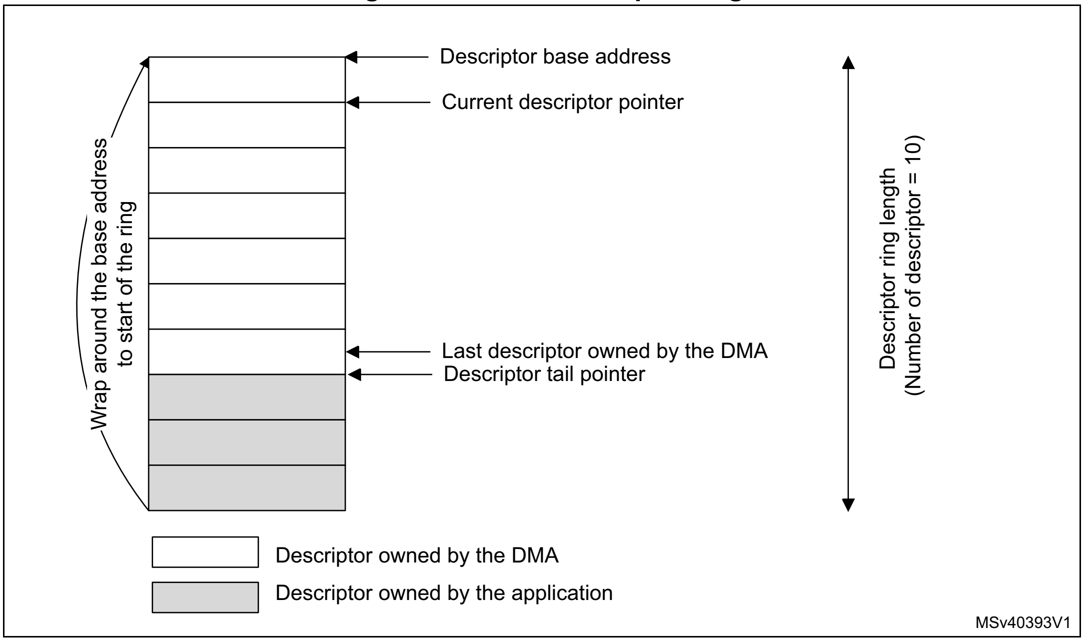
```
Descriptor Tail Pointer > Current Descriptor Pointer;
```

Note: When the application sets the descriptors into the end-of-descriptor ring, continues setting from the start of the descriptor list and updating the descriptor tail pointer, the above condition is not satisfied. The DMA detects such a case, processes the descriptors until the end of the ring and then continues processing from the start of the descriptor list until the current descriptor pointer is equal to the descriptor tail pointer.

When the application programs the descriptor tail pointer beyond the end of the descriptor ring, the DMA interprets this as the availability of one more complete ring of descriptors, and after processing the last descriptor in the ring continues to process descriptors from the start of the descriptor list. If this is not intended, the application must program the descriptor tail pointer to the start of the descriptor list after setting the last descriptor in the ring.

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 762: DMA descriptor ring*.

Figure 762. DMA descriptor ring



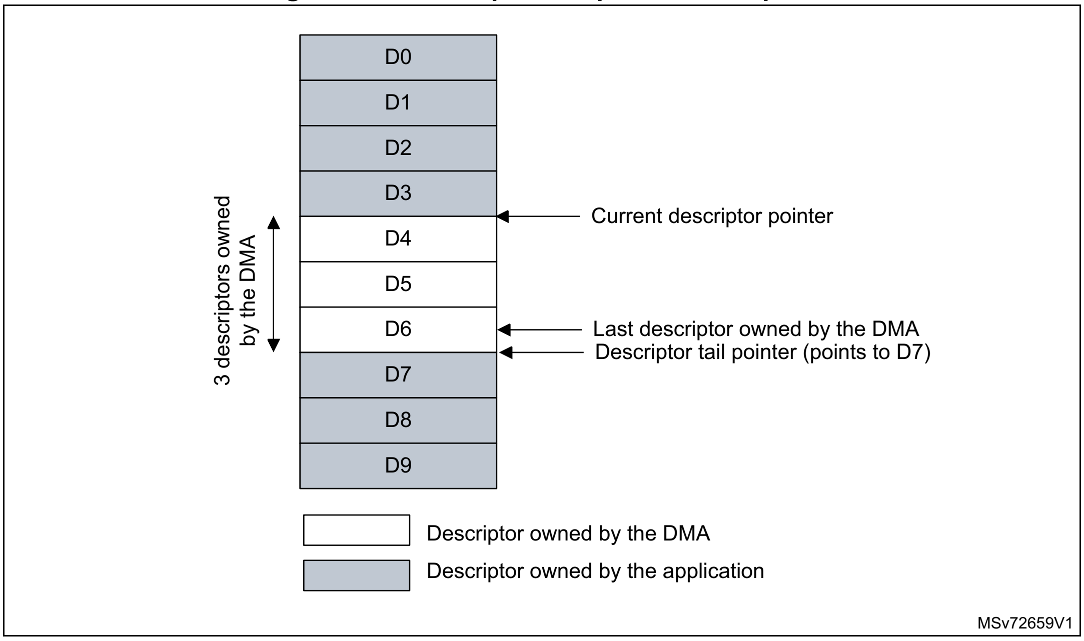
For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.

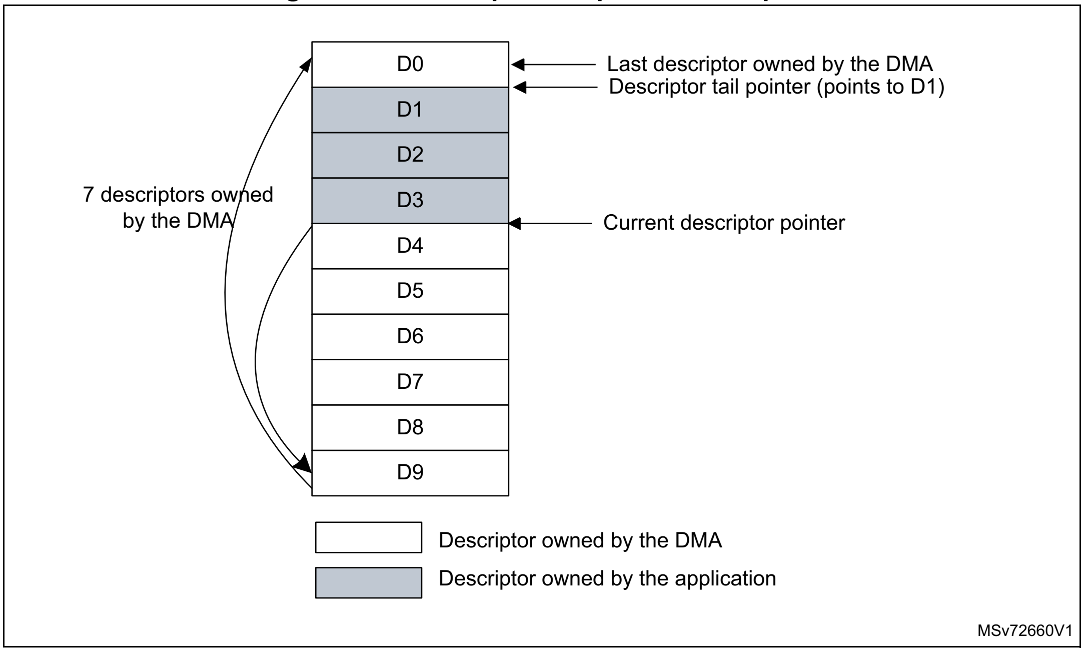
Descriptor tail pointer handling examples

Figure 763. Descriptor tail pointer example 1



When the current descriptor pointer points to descriptor 4 (D4) and the descriptor tail pointer points to descriptor 7 (D7), the last descriptor owned by the DMA is descriptor 6 (D6) and there are three descriptors (D4, D5, D6) available for the DMA, as shown in *Figure 763: Descriptor tail pointer example 1*.

Figure 764. Descriptor tail pointer example 2



As shown in *Figure 764: Descriptor tail pointer example 2*, the application frees four descriptors (D7, D8, D9 and D0) and updates the descriptor tail pointer to point to descriptor 1 (D1). The last descriptor owned by the DMA is descriptor 0 (D0). The descriptors referenced between the current descriptor pointer (D4) and the descriptor tail pointer (D1) are available to the DMA.

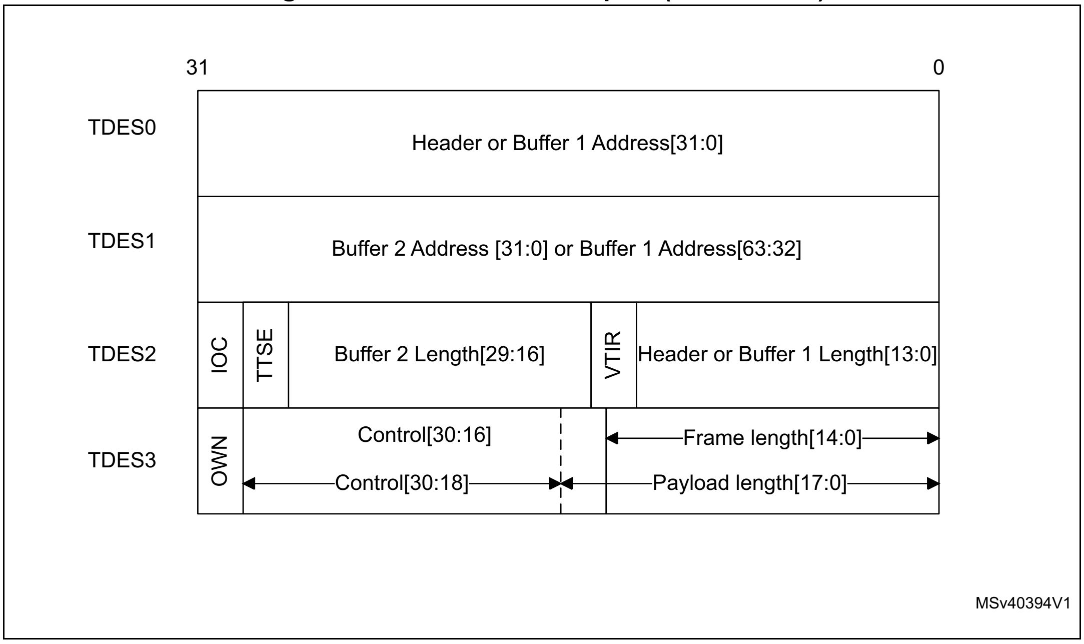
61.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format

Transmit normal descriptor (read format)

Figure 765 shows the Read format for Transmit normal descriptor. *Table 513* to *Table 516* provide a detailed description of all Transmit normal descriptors (read format).

Figure 765. Transmit descriptor (read format)



- TDES0 normal descriptor (read format)

Table 513. TDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer or TSO Header Address Pointer** These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set: – TSE bit of TDES3 – FD bit of TDES3

- TDES1 normal descriptor (read format)

Table 514. TDES1 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 or Buffer 1 Address Pointer:** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment.

- TDES2 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IOC	TTSE	B2L													
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VTIR		HL or B1L													

Table 515. TDES2 normal descriptor (read format)

Bits	Name	Description
31	IOC	**Interrupt on completion:** This bit sets the TI bit in the *Channel x status register (ETH_DMACxSR)*) when the present packet transmission is complete.
30	TTSE	**Transmit Timestamp Enable** This bit enables the IEEE1588 timestamping for Transmit packet referenced by the descriptor.
29:16	B2L	**Buffer 2 Length** The driver sets this field. When set, this field indicates Buffer 2 length.
15:14	VTIR	**VLAN Tag Insertion or Replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows: 00: Do not add a VLAN tag. 01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10: Insert a VLAN tag with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. 11: Replace the VLAN tag in packets with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. This option should be used only with the VLAN packets.

Table 515. TDES2 normal descriptor (read format) (continued)

Bits	Name	Description
13:0	HL or B1L	**Header length or buffer 1 length** For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.

- TDES3 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	CPC		SAIC			SLOTNUM				TSE	TPL	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FT/L															

Table 516. TDES3 normal descriptor (read format)

Bits	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value.

Table 516. TDES3 normal descriptor (read format) (continued)

Bits	Name	Description
27:26	CPC	**CRC Pad Control** This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following: 00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	**SA Insertion Control** These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[24:23]: 00: Do not include the source address 01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11: Reserved These bits are valid when the First Segment control bit (TDES3 [29]) is set.

Table 516. TDES3 normal descriptor (read format) (continued)

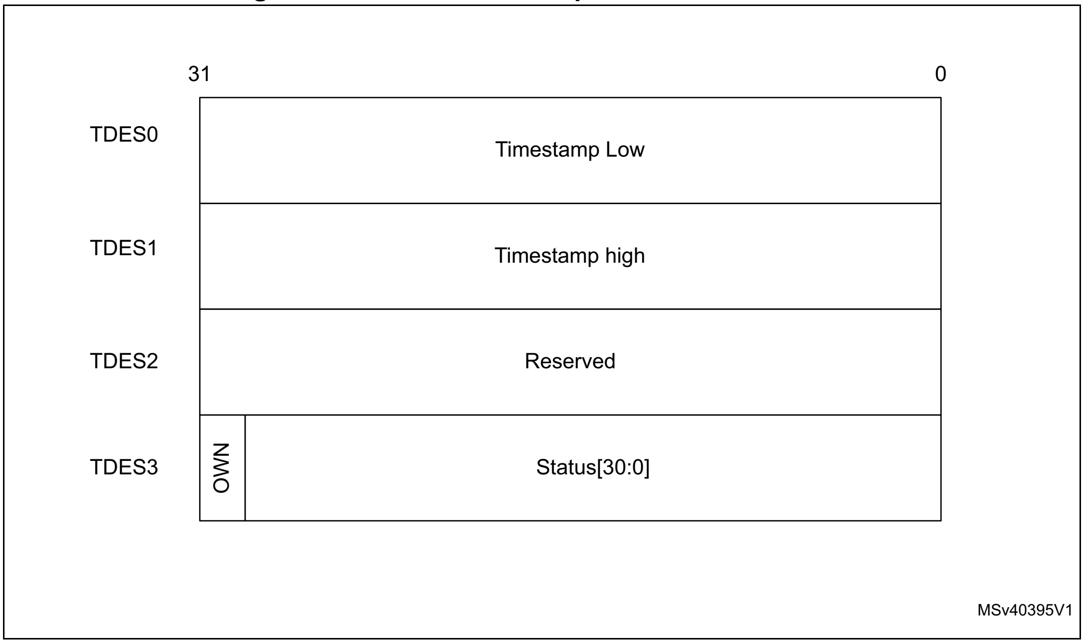
Bits	Name	Description
22:19	SLOTNUM or THL	**SLOTNUM**: Slot Number Control Bits in AV mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field of *Channel x slot function control status register (ETH_DMACxSFCSR)*. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels. **THL**: TCP Header Length If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor
18	TSE	**TCP Segmentation Enable** When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set.
17:16	CIC/TPL	**Checksum Insertion Control or TCP Payload Length** These bits control the checksum calculation and insertion. They can take the following values: 00: Checksum insertion disabled. 01: Only IP header checksum calculation and insertion are enabled. 10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11: IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the TSE bit is reset.When the TSE bit is set, it contains the upper bits [17:16] of the TCP Payload length. This allows the TCP packet length field to be spanned across TDES3[17:0] to provide 256 Kbyte packet length support.
15	TPL	**Reserved or TCP Payload Length** When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0].
14:0	FL/TPL	**Reserved or TCP Payload Length** When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/UDP/IP header length. When the TSE bit is reset, this bit is reserved.

Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 766 shows the write-back format for Transmit normal descriptors. *Table 517* to *Table 520* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

Figure 766. Transmit descriptor write-back format



- TDES0 normal descriptor (write-back format)

Table 517. TDES0 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES1 normal descriptor (write-back format)

Table 518. TDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)

Table 519. TDES2 normal descriptor (write-back format)(1)

Bit	Description
31:0	Reserved

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	Reserved										TTSS	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ES	JT	FF	PCE	LoC	NC	LC	EC	CC				ED	UF	DB	IHE

Table 520. TDES3 normal descriptor (write-back format)(1)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptors.
29	FD	**First Descriptor** This bit indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:18	Reserved	
17	TTSS	**Tx Timestamp Status** This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.
16	Reserved	

Table 520. TDES3 normal descriptor (write-back format)[1] (continued)

Bit	Name	Description
15	ES	**Error Summary** This bit indicates the logical OR of the following bits: TDES3[0]: IP Header Error TDES3[14]: Jabber Timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error
14	JT	**Jabber Timeout** This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the *Operating mode configuration register (ETH_MACCR)* is not set.
13	FF	**Packet Flushed** This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	**Payload Checksum Error** This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.
11	LoC	**Loss of Carrier** This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.
10	NC	**No Carrier** This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	**Late Collision** This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode and 512 byte times including Preamble and Carrier Extension in GMII mode). This bit is not valid if Underflow Error is set.

Table 520. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
8	EC	**Excessive Collision** This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	**Collision Count** This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	**Excessive Deferral** This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the *Operating mode configuration register (ETH_MACCR)*.
2	UF	**Underflow Error** This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register.
1	DB	**Deferred Bit** This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode.
0	IHE	**IP Header Error** When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

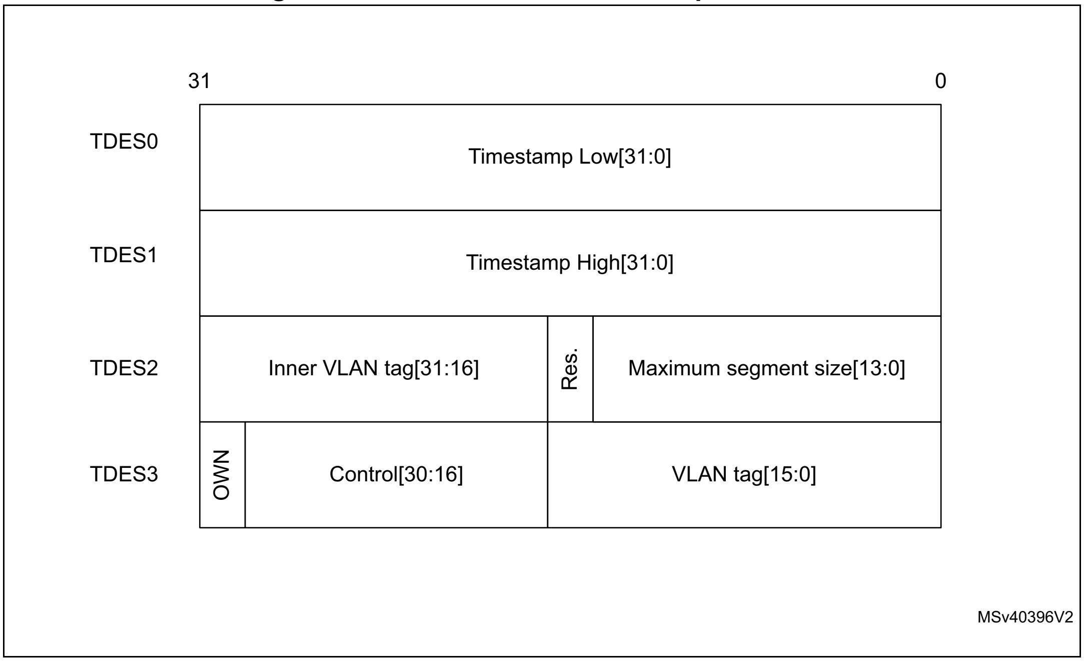
Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 767 shows the format for Transmit context descriptors. Table 521 to Table 524 provide a detailed description of all Transmit context descriptors.

Figure 767. Transmit context descriptor format



- TDES0 context descriptor (read format)

Table 521. TDES0 context descriptor

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)

Table 522. TDES1 context descriptor

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES2 context descriptor (read format)

Table 523. TDES2 context descriptor

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	Reserved	
13:0	MSS	**Maximum Segment Size** This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

- TDES3 context descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	Reserved		OSTC	TCMSSV	Reserved		CDE	Reserved					IVLTV	VLTV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VT															

Table 524. TDES3 context descriptor

Bit	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit immediately after a read operation.
30	CTXT	**Context Type** This bit should be set to 1 for context descriptor.
29:28	Reserved	
27	OSTC	**One-Step Timestamp Correction Enable** When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.

Table 524. TDES3 context descriptor (continued)

Bit	Name	Description
26	TCMSSV	**One-Step Timestamp Correction Input or MSS Valid** When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Reserved	
23	CDE	**Context Descriptor Error** When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor. The Context Descriptor errors can be: – Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. – All 1s. – CD, LD, and FD bits set to 1. *Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel x status register (ETH_DMACxSR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.* *This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported)*
22:20	Reserved	
19:18	IVTIR	Inner VLAN Tag Insert or Replace When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. This bitfield has the following values: 00: Do not add the inner VLAN tag. 01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10: Insert an inner VLAN tag with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. 11: Replace the inner VLAN tag in packets with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. This option should be used only with the VLAN frames.
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.

Table 524. TDES3 context descriptor (continued)

Bit	Name	Description
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the *VLAN inclusion register (ETH_MACVIR)* is reset.

61.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following Receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

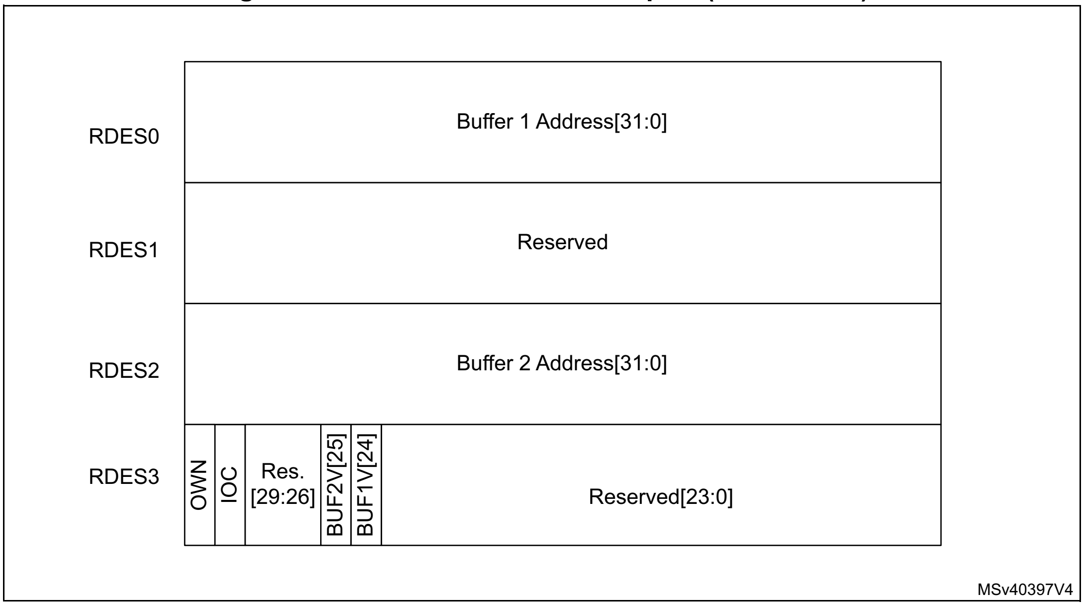
All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see *Figure 768: Receive normal descriptor (read format)* for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in *Figure 769: Receive normal descriptor (write-back format)*.

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in *Figure 770: Receive context descriptor*.

Receive normal descriptor (read format)

Figure 768 shows the read format for Receive normal descriptors. *Table 525* to *Table 528* provide a detailed description of all Receive normal descriptors (read format).

Figure 768. Receive normal descriptor (read format)



Note: In the Receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.

- RDES0 normal descriptor (read format)

Table 525. RDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES2[2:0]=0 in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

- RDES1 normal descriptor (read format)

Table 526. RDES1 normal descriptor (read format)

Bit	Name	Description
31:0	Reserved	Field reserved.

- RDES2 normal descriptor (read format)

Table 527. RDES2 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 Address Pointer** These bits indicate Buffer 2 physical address. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[2:0]=0 and writes to the complete location.

- RDES3 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IOC	Reserved				BUF2V	BUF1V	Reserved							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

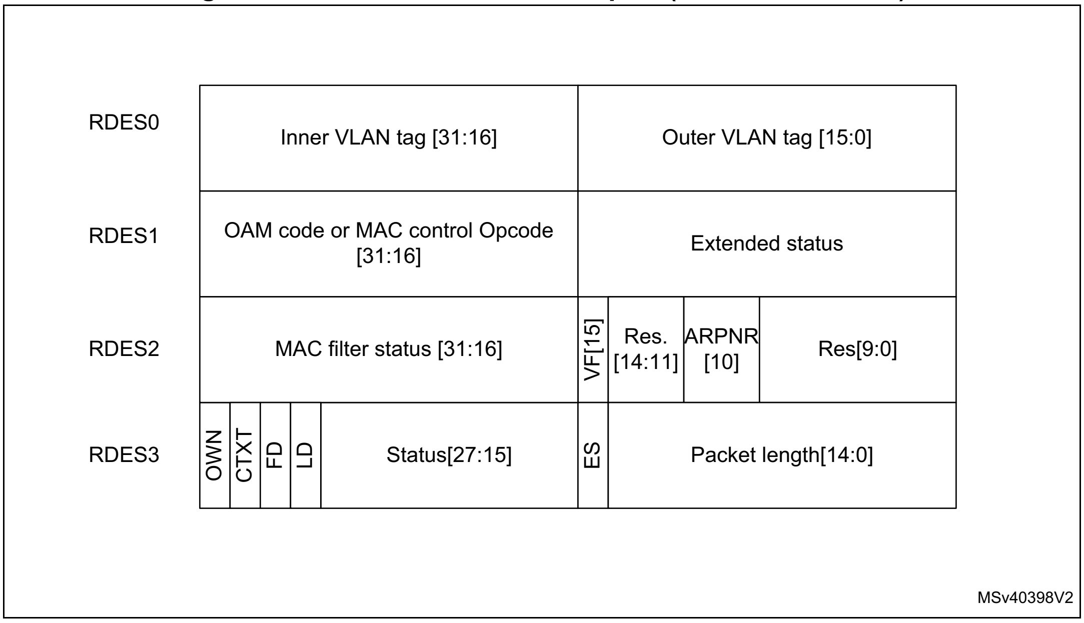
Table 528. RDES3 normal descriptor (read format)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	IOC	**Interrupt Enabled on Completion** When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Reserved	
25	BUF2V	**Buffer 2 Address Valid** When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.
24	BUF1V	**Buffer 1 Address Valid** When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.
23:0	Reserved	

Receive normal descriptor (write-back format)

Figure 769 shows the write-back format for Receive normal descriptors. *Table 529* to *Table 532* provide a detailed description of all Receive normal descriptors (write-back format).

Figure 769. Receive normal descriptor (write-back format)



- RDES0 normal descriptor (write-back format)

Table 529. RDES0 normal descriptor (write-back format)

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** This field contains the Inner VLAN tag of the received packet if the RSOV bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	**Outer VLAN Tag** This field contains the Outer VLAN tag of the received packet if the RSOV bit of RDES3 is set.

- RDES1 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPC															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TD	TSA	PV	PFT	PMT				IPCE	IPCB	IPV6	IPV4	IPHE	PT		

Table 530. RDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:16	OPC	**OAM Subtype Code, or MAC Control Packet opcode** OAM Subtype Code If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields. MAC Control Packet opcode If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
15	TD	**Timestamp Dropped** This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.
14	TSA	**Timestamp Available** When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	**PTP Version** 1: Received PTP message in IEEE 1588 version 2 format 0: Received PTP message in IEEE 1588 version 1 format
12	PFT	**PTP Packet Type** This bit indicates that the PTP message is sent directly over Ethernet.
11:8	PMT	**PTP Message Type** These bits are encoded to give the type of the message received: 0000: No PTP message received 0001: SYNC (all clock types) 0010: Follow_Up (all clock types) 0011: Delay_Req (all clock types) 0100: Delay_Resp (all clock types) 0101: Pdelay_Req (in peer-to-peer transparent clock) 0110: Pdelay_Resp (in peer-to-peer transparent clock) 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000: Announce 1001: Management 1010: Signaling 1011–1110: Reserved 1111: PTP packet with Reserved message type
7	IPCE	**IP Payload Error** When this bit is set, it indicates either of the following: – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.

Table 530. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
6	IPCB	**IP Checksum Bypassed** This bit indicates that the checksum offload engine is bypassed.
5	IPV6	**IPv6 header Present** This bit indicates that an IPV6 header is detected.
4	IPV4	**IPV4 Header Present** This bit indicates that an IPV4 header is detected.
3	IPHE	**IP Header Error** – When this bit is set, it indicates either of the following: – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. – The IP datagram version is not consistent with the Ethernet Type value. – Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
2:0	PT	**Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): 000: Unknown type or IP/AV payload not processed 001: UDP 010: TCP 011: ICMP 100: IGMP if IPV4 Header Present bit is set 110: AV Tagged Data Packet 111: AV Tagged Control Packet 101: AV Untagged Control Packet Others: reserved. If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- RDES2 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3L4FM			L4FM	L3FM	MADRM								HF	DAF	SAF
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VF	Reserved				ARPRN	Reserved									

Table 531. RDES2 normal descriptor (write-back format)

Bit	Name	Description
31:29	L3L4FM	**Layer 3 and Layer 4 Filter Number Matched** These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: – 000: Filter 0 – 001: Filter 1 – 010: Filter 2 – 011: Filter 3 – 100: Filter 4 – 101: Filter 5 – 110: Filter 6 – 111: Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter.
28	L4FM	**Layer 4 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: – Layer 3 fields are not enabled and all enabled Layer 4 fields match – All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29].
27	L3FM	**Layer 3 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: – All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed – All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29].
26:19	MADRM	**MAC Address Match or Hash Value** When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register.
18	HF	**Hash Filter Status** When this bit is set, it indicates that the packet passed the MAC address Hash filter. its[26:19] indicate the Hash value.
17	DAF	**Destination Address Filter Fail** When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
16	SAF	**SA Address Filter Fail** When this bit is set, it indicates that the packet failed the SA Filter in the MAC.

Table 531. RDES2 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	VF	**VLAN Filter Status** When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter.
14:11	Reserved	
10	ARPNR	**ARP Reply Not Generated** When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
9:0	Reserved	

- RDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ES	PL														

Table 532. RDES3 normal descriptor (write-back format)

Bit	Name	Description
31	OWN	**Own bit** 1: The DMA owns the descriptor. 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} possible values are: 00: Intermediate Descriptor 01: First Descriptor 10: Reserved 11: Descriptor error (due to all 1s) *Note: When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel x status register (ETH_DMACxSR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*

Table 532. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
29	FD	**First Descriptor** When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	**Receive Status RDES2 Valid** When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	**Receive Status RDES1 Valid** When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	**Receive Status RDES0 Valid** When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
24	CE	**CRC Error** When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	**Giant Packet** When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	**Receive Watchdog Timeout** When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	**Overflow Error** When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.

Table 532. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
20	RE	**Receive Error** When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception. This error also includes carrier extension error in the GMII and Half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	**Dribble Bit Error** When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	**Length/Type Field** This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000: The packet is a length packet 001: The packet is a type packet. 011: The packet is a ARP Request packet type 100: The packet is a type packet with VLAN Tag 101: The packet is a type packet with Double VLAN tag 110: The packet is a MAC Control packet type 111: The packet is a OAM packet type 010: Reserved
15	ES	**Error Summary** When this bit is set, it indicates the logical OR of the following bits: RDES3[19]: Dribble Error RDES3[20]: Receive Error RDES3[21]: Overflow Error RDES3[22]: Watchdog Timeout RDES3[23]: Giant Packet RDES3[24]: CRC Error This field is valid only when the LD bit of RDES3 is set.
14:0	PL	**Packet Length** These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet.

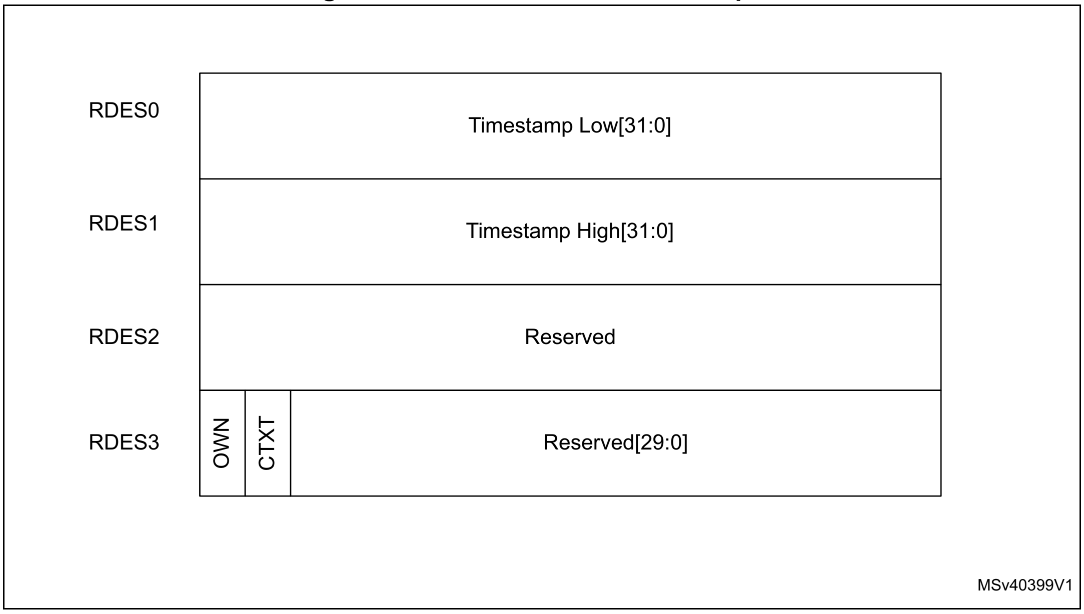
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 770 shows the format for Receive context descriptors. *Table 533* to *Table 536* provide a detailed description of all Receive context descriptors.

Figure 770. Receive context descriptor



- RDES0 context descriptor

Table 533. RDES0 context descriptor

Bit	Name	Description
31:0	RTSL	**Receive Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

- RDES1 context descriptor

Table 534. RDES1 context descriptor

Bit	Field	Description
31:0	RTSH	**Receive Packet Timestamp High** The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

- RDES2 context descriptor

Table 535. RDES2 context descriptor

Bit	Description
31:0	Reserved

- RDES3 context descriptor

Table 536. RDES3 context descriptor

Bit	Name	Description
31	OWN	**Own Bit** 1: The DMA owns the descriptor 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.
29:0	Reserved	

61.11 Ethernet registers

61.11.1 Ethernet register maps

This section provides the following register maps:

- DMA registers (see *Section 61.11.2: Ethernet DMA registers*)
- MTL registers (see *Section 61.11.3: Ethernet MTL registers*)
- MAC registers including MMC registers (see *Section 61.11.4: Ethernet MAC and MMC registers*)

61.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000

Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TXPR	Res.	Res.	DSPW	Res.	Res.	Res.	TAA2	TAA1	TAA0	Res.	SWR
				rw			rw				r	r	r		rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **INTM[1:0]**: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.

The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to *Table 512: Transfer complete interrupt behavior*).

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TXPR**: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **DSPW**: Descriptor Posted Write

When this bit is set to

0: The descriptor writes are always non-posted.

1: The descriptor writes are non-posted only when IOC (Interrupt on completion) is set in last descriptor. Otherwise, the descriptor writes are always posted.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:2 **TAA[2:0]**: Transmit Arbitration Algorithm

This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected.

000: Fixed priority. In fixed priority, Channel 0 has the lowest priority and the last channel has the highest priority.

001: Weighted Strict priority (WSP)

010: Weighted Round-Robin (WRR)

011 to 111: Reserved, must not be used.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.

System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0101 0000

The System bus mode register controls the behavior of the AXI master. It mainly controls burst splitting and number of outstanding requests.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_LPI	LPI_XIT_PKT	Res.	Res.	Res.	Res.	WR_OSR_LMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]	
rw	rw					rw	rw							rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	ONEKBBE	AAL	Res.	AALE	Res.	Res.	BLEN256	BLEN128	BLEN64	BLEN32	BLEN16	BLEN8	BLEN4	FB
		rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EN_LPI**: Enable Low Power Interface (LPI)

When set to 1, this bit enables the LPI mode and accepts the LPI request from the AXI System Clock controller.

When set to 0, this bit disables the LPI mode and always denies the LPI request from the AXI System Clock controller.

Bit 30 **LPI_XIT_PKT**: Unlock on Magic Packet or Remote wake-up Packet

When set to 1, this bit enables the AXI master to come out of the LPI mode only when the magic packet or remote wake-up packet is received. When set to 0, this bit enables the AXI master to come out of the LPI mode when any packet is received.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **WR_OSR_LMT[1:0]**: AXI Maximum Write Outstanding Request Limit

This value limits the maximum outstanding request on the AXI write interface. Maximum outstanding requests = WR_OSR_LMT + 1.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **RD_OSR_LMT[1:0]**: AXI Maximum Read Outstanding Request Limit

This value limits the maximum outstanding request on the AXI read interface. Maximum outstanding requests = RD_OSR_LMT + 1

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ONEKBBE**: 1 Kbyte Boundary Crossing Enable for the AXI Master

When set, the burst transfers performed by the AXI master do not cross 1 Kbyte boundary. When reset, the burst transfers performed by the AXI master do not cross 4 Kbyte boundary.

Bit 12 **AAL**: Address-Aligned Beats

When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **AALE**: Automatic AXI LPI enable

When set to 1, enables the AXI master to enter into LPI state when there is no activity in the Ethernet peripheral for a number of system clock cycles programmed in the LPIEI field of ETH_DMALPIEI register.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **BLEN256**: AXI Burst Length 256

When this bit is set to 1, the AXI master can select a burst length of 256 on the AXI interface.

Bit 6 **BLEN128**: AXI Burst Length 128

When this bit is set to 1, the AXI master can select a burst length of 128 on the AXI interface.

Bit 5 **BLEN64**: AXI Burst Length 64

When this bit is set to 1, the AXI master can select a burst length of 64 on the AXI interface.

Bit 4 **BLEN32**: AXI Burst Length 32

When this bit is set to 1, the AXI master can select a burst length of 32 on the AXI interface.

Bit 3 **BLEN16**: AXI Burst Length 16

When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 16 on the AXI interface.

When the FB bit is set to 0, setting this bit has no effect.

Bit 2 **BLEN8**: AXI Burst Length 8

When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 8 on the AXI interface.

When the FB bit is set to 0, setting this bit has no effect.

Bit 1 **BLEN4**: AXI Burst Length 4

When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 4 on the AXI interface.

When the FB bit is set to 0, setting this bit has no effect.

Bit 0 **FB**: Fixed Burst Length

When this bit is set to 1, the AXI master initiates burst transfers of specified lengths as given below.

– Burst transfers of fixed burst lengths as indicated by the BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, or BLEN4 field
– Burst transfers of length 1: when this bit is set to 0, the AXI master initiates burst transfers that are equal to or less than the maximum allowed burst length programmed in Bits[7:1].

Interrupt status register (ETH_DMAISR)

Address offset: 0x1008

Reset value: 0x0000 0000

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC1IS	DC0IS
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **MACIS**: MAC Interrupt Status

This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL Interrupt Status

This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **DC1IS**: DMA Channel 1 Interrupt Status

This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source.

Bit 0 **DC0IS**: DMA Channel 0 Interrupt Status

This bit indicates an interrupt event in DMA Channel 0. To reset this bit to 0, the software must read the corresponding register in DMA Channel 0 to get the exact cause of the interrupt and clear its source.

Debug status register (ETH_DMADSR)

Address offset: 0x100C

Reset value: 0x0000 0000

The Debug status register gives the Receive and Transmit process status for DMA Channel 0 to 1 for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS1[3:0]				RPS1[3:0]			
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	AXRH STS	AXWH STS
r	r	r	r	r	r	r	r							r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **TPS1[3:0]**: DMA Channel 1 Transmit Process State
This field indicates the Tx DMA FSM state for Channel 1. This field is similar to the TPS0 field.

Bits 19:16 **RPS1[3:0]**: DMA Channel 1 Receive Process State
This field indicates the Rx DMA FSM state for Channel 1. This field is similar to the RPS0 field.

Bits 15:12 **TPS0[3:0]**: DMA Channel 0 Transmit Process State
This field indicates the Tx DMA FSM state for Channel 0:
000: Stopped (Reset or Stop Transmit Command issued)
001: Running (Fetching Tx Transfer Descriptor)
010: Running (Waiting for status)
011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
100: Timestamp write state
101: Reserved for future use
110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA Channel 0 Receive Process State
This field indicates the Rx DMA FSM state for Channel 0:
000: Stopped (Reset or Stop Receive Command issued)
001: Running (Fetching Rx Transfer Descriptor)
010: Reserved for future use
011: Running (Waiting for Rx packet)
100: Suspended (Rx Descriptor Unavailable)
101: Running (Closing the Rx Descriptor)
110: Timestamp write state
111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **AXRHSTS**: AXI Master Read Channel Status
When high, this bit indicates that the read channel of the AXI master is active, and it is transferring the data.

Bit 0 **AXWHSTS**: AXI Master Write Channel
When high, this bit indicates that the write channel of the AXI master is active, and it is transferring data.

AXI4 transmit channel ACE control register (ETH_DMAA4TXACR)

Address offset: 0x1020

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for read transactions by all the Transmit DMA channels. The following signals of the AXI4 interface are driven with different values as programmed for corresponding type (descriptor, buffer1, buffer2) of access:

- arcache_m_o[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THC[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TEC[3:0]				Res.	Res.	Res.	Res.	TDRC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **THC[3:0]**: Transmit DMA First Packet Buffer or TSO Header Cache Control
When TSO is NOT enabled, This field is used to drive arcache_o[3:0] signal when Transmit DMA is accessing First Buffer of the Packet (First valid buffer with FD being set in the TDES3 of the Descriptor).
When TSO is enabled, This field is used to drive arcache_o[3:0] signal when the Transmit DMA is accessing the TSO Header data.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TEC[3:0]**: Transmit DMA Extended Packet Buffer or TSO Payload Cache Control
When TSO is NOT enabled, This field is used to drive arcache o[3:0] signal when Transmit DMA is accessing the extended buffers (when packet is distributed across multiple buffers).
When TSO is enabled, This field is used to drive arcache_o[3:0] signal when the Transmit DMA is accessing the TSO payload data.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TDRC[3:0]**: Transmit DMA Read Descriptor Cache Control
This field is used to drive arcache_o[3:0] signal when Transmit DMA engines access the Descriptor.

AXI4 receive channel ACE control register (ETH_DMAA4RXACR)

Address offset: 0x1024

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for write transactions by all the Receive DMA channels. The following signals of the AXI4 interface are driven with different values as programmed for corresponding type (descriptor, buffer1, buffer2) of access:

- awcache_m_o[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RDC[3:0]				Res.	Res.	Res.	Res.	RHC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RPC[3:0]				Res.	Res.	Res.	Res.	RDWC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **RDC[3:0]**: Receive DMA Buffer Cache Control
This field is used to drive awcache_o[3:0] signal when Receive DMA is accessing the Buffer when Header and payload are NOT separated.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **RHC[3:0]**: Receive DMA Header Cache Control
This field is used to drive awcache_o[3:0] and signal when Receive DMA is accessing the header Buffer when Header and payload are separated.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **RPC[3:0]**: Receive DMA Payload Cache Control
This field is used to drive awcache_o[3:0] signal when Receive DMA is accessing the Payload Buffer when Header and payload are separated.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **RDWC[3:0]**: Receive DMA Write Descriptor Cache Control
This field is used to drive awcache_o[3:0] signal when Receive DMA accesses the Descriptor.

AXI4 descriptor ACE control register (ETH_DMAA4DACR)

Address offset: 0x1028

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for Descriptor write transactions by all the TxDMA channels and Descriptor read transactions by all the RxDMA channels. It also controls the values to be driven on awprot_m_o and arprot_m_o.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RDRC[3:0]				Res.	Res.	TDWD[1:0]		TDWC[3:0]			
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **RDRC[3:0]**: Receive DMA Read Descriptor Cache control
This field is used to drive arcache_o[3:0] signal when Receive DMA engines read the Descriptor.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **TDWD[1:0]**: Transmit DMA Write Descriptor Domain control
This field is used to drive awdomain_o[1:0] signal when Transmit DMA write to the Descriptor.

Bits 3:0 **TDWC[3:0]**: Transmit DMA Write Descriptor Cache control
This field is used to drive awcache_o[3:0] signal when Transmit DMA writes to the Descriptor.

AXI4 LPI Entry Interval register (ETH_DMALPIEI)

Address offset: 0x1040

Reset value: 0x0000 0000

This register is used to control the AXI LPI entry interval.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIEI[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **LPIEI[3:0]**: LPI Entry Interval

Contains the number of system clock cycles, multiplied by 64, to wait for an activity in the peripheral to enter into the AXI low power state.

0 indicates 64 clock cycles.

Channel x control register (ETH_DMACxCR)

Address offset: 0x1100 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The DMA Channel x control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8
											rw	rw	rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	MSS[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **DSL[2:0]**: Descriptor Skip Length

This bit specifies the 64-bit double-word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.

When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PBLX8**: 8xPBL mode

When this bit is set, the PBL value programmed in Bits[21:16] in *Channel x transmit control register (ETH_DMACxTXCR)* and in Bits[21:16] in *Channel 0 receive control register (ETH_DMAC0RXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **MSS[13:0]**: Maximum Segment Size

This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel x transmit control register (ETH_DMACxTXCR)* is set.

The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

Channel x transmit control register (ETH_DMACxTXCR)

Address offset: 0x1104 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The DMA Channel x Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TQOS[3:0]				Res.	Res.	TXPBL[5:0]					
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
rw			rw								rw	r	r	r	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bits 27:24 **TQOS[3:0]**: Transmit QOS

This field is used to drive arqos_m_o[3:0] or awqos_m_o[3:0] output signals for all transactions of DMA Tx Channel x.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TXPBL[5:0]**: Transmit Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in ETH_DMACxCR.

b) Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue x operating mode register (ETH_MTLTXQxOMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 4096 bytes is 512, TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 **IPBL**: Ignore PBL Requirement

When this bit is set, the DMA does not check for PBL number of locations in the MTL before initiating a transfer. If space is not available, the MTL might use handshaking to slow the DMA.

Note: This bit/mode must not be used when multiple Transmit DMA Channels are enabled as it might block other Transmit and Receive DMA Channels from accessing the Read Data Channel of AXI bus until space is available in Transmit Queue for current transfer.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TSE**: TCP Segmentation Enabled

When this bit is set, the DMA performs the TCP segmentation for packets in Channel x. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF**: Operate on Second Packet

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 **TCW[2:0]**: Transmit Channel Weight

This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.

Bit 0 **ST**: Start or Stop Transmission Command

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the base address of the Transmit list set by the ETH_DMACxTXDLAR register.

– The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACxSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACxTXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACxTXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

Channel 0 receive control register (ETH_DMAC0RXCR)

Address offset: 0x1108

Reset value: 0x0000 0000

The DMA Channel 0 Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPF	Res.	Res.	Res.	RQOS[3:0]				Res.	Res.	RXPBL[5:0]					
rw				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RBSZ[13:0]														SR
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RPF**: DMA Rx Channel 0Packet Flush

When this bit is set to 1, the Ethernet peripheral automatically flushes the packet from the Rx queues destined to DMA Rx Channel 0 when the DMA Rx Channel 0 is stopped after a system bus error has occurred. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing happens on the Read side of the Rx queue.

When this bit is set to 0 the Ethernet peripheral does not flush the packet in the Rx queue destined to DMA Rx Channel 0 after the DMA is stopped due to a system bus error.

This might cause head-of-line blocking in the corresponding RxQueue.

Note: The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in ETH_DMAC0RXCR register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **RQOS[3:0]**: Rx AXI4 QOS.

This field is used to drive arqos_m_o[3:0] or awqos_m_o[3:0] output signals for all transactions of DMA Rx Channel0.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in the ETH_DMAC0CR.

b) Set the RXPBL[5:0].

Note: The maximum value of RXPBL must be less than or equal to half the Rx Queue size (RQS field of Rx queue x operating mode register (ETH_MTLRXQxOMR)) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the MTL Rx Queue Controller is transferring data to MAC. The total locations in Rx Queue of size 4096 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 Reserved, must be kept at reset value.

Bits 14:1 **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

Note: The buffer size must be a multiple of 8. This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 8, it may result into undefined behavior.

The LSB bits (2:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0 **SR**: Start or Stop Receive

When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the address set by the *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*.

– The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMAC0SR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the *Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)*, the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)

Address offset: 0x1114 +0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Channel x Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Dword-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in ETH_DMACxTXCR register. When stopped, this register can be written with a new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **TDESLA[31:0]**: Start of Transmit List
This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (2:0) for 64-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel 0 Rx descriptor list address register (ETH_DMAC0RXDLAR)

Address offset: 0x111C

Reset value: 0x0000 0000

The Channel 0 Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Dword-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMAC0RXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **RDESLA[31:0]**: Start of Receive List
This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (2:0) for 64-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)

Address offset: 0x1120 + 0x80 * x (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **TDT[31:0]**: Transmit Descriptor Tail Pointer

This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

Channel 0 Rx descriptor tail pointer register (ETH_DMAC0RXDTPR)

Address offset: 0x1128

Reset value: 0x0000 0000

The Channel 0 Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **RDT[31:0]**: Receive Descriptor Tail Pointer

This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)

Address offset: 0x112C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TDRL[9:0]**: Transmit Descriptor Ring Length

This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It is recommended to put a minimum ring descriptor length of 4.

For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

Channel 0 Rx descriptor ring length register (ETH_DMAC0RXRLR)

Address offset: 0x1130

Reset value: 0x0000 0000

The Channel 0 Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ARBS[6:0]**: Alternate Receive Buffer Size

Indicates size in bytes for Buffer 1 when ARBS[6:0] is programmed to a non-zero value.

When ARBS[6:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of *Channel 0 receive control register (ETH_DMAC0RXCR)*.

Bits 16:10 Reserved, must be kept at reset value.

Bits 9:0 **RDRL[9:0]**: Receive Descriptor Ring Length

This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors.

For example, you can program any value up to 0x3FF in this field. This field is 10-bit wide. If you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

Channel x interrupt enable register (ETH_DMACxIER)

Address offset: 0x1134 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Interrupt Enable register enables the interrupts reported by the Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable

When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the *Channel x status register (ETH_DMACxSR)*:

Bit 0: Transmit interrupt

Bit 2: Transmit buffer unavailable

Bit 6: Receive interrupt

Bit 11: Early Receive interrupt

When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable

When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the *Channel x status register (ETH_DMACxSR)*:

Bit 1: Transmit Process Stopped

Bit 7: Rx Buffer Unavailable

Bit 8: Receive Process Stopped

Bit 9: Receive Watchdog Timeout

Bit 10: Early transmit interrupt

Bit 12: Fatal Bus Error

When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context Descriptor Error Enable

When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error Enable

When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early receive interrupt Enable

When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.

Bit 10 **ETIE**: Early Transmit Interrupt Enable

When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive Watchdog Timeout Enable

When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.

Bit 8 **RSE**: Receive Stopped Enable

When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 **RBUE**: Receive Buffer Unavailable Enable

When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.

Bit 6 **RIE**: Receive Interrupt Enable

When this bit is set along with the NIE bit, the receive interrupt is enabled. When this bit is reset, the receive interrupt is disabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable Enable

When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

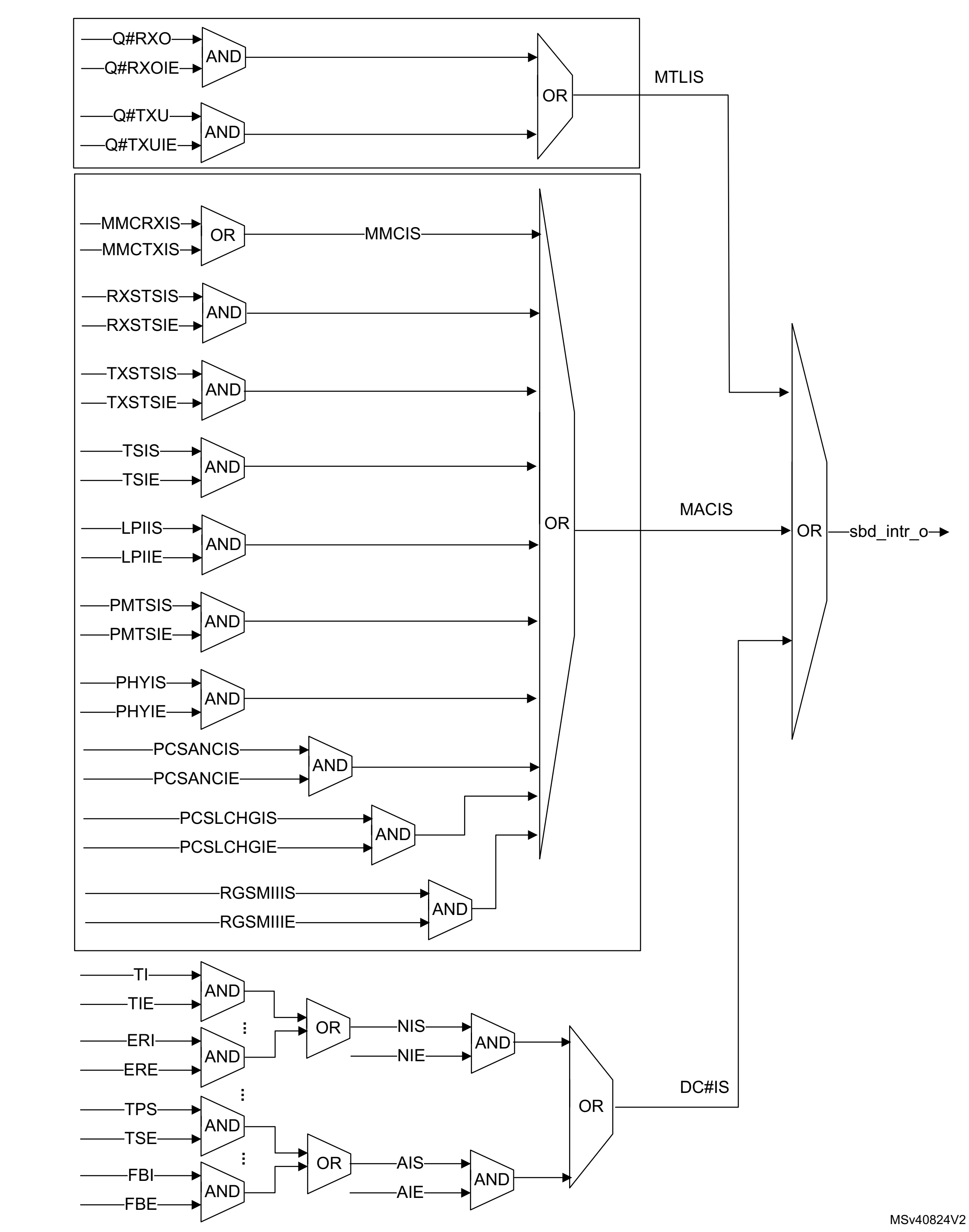
Bit 1 **TXSE**: Transmit Stopped Enable

When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt Enable

When this bit is set along with the NIE bit, the transmit interrupt is enabled. When this bit is reset, the transmit interrupt is disabled.

Figure 771. Generation of ETH_DMAISR flags



Channel 0 Rx interrupt watchdog timer register (ETH_DMAC0RXIWTR)

Address offset: 0x1138

Reset value: 0x0000 0000

The receive interrupt Watchdog Timer register indicates the watchdog timeout for receive interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the *Channel x status register (ETH_DMACxSR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]	
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **RWTU[1:0]**: Receive Interrupt Watchdog Timer Count Units

This bitfield indicates the number of system clock cycles corresponding to one unit in RWT[7:0] bitfield.

00: 256

01: 512

10: 1024

11: 2048

For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 × 512 = 1024 system clock cycles.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This bitfield indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.

The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMAC0SR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].

When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

Channel x slot function control status register (ETH_DMACxSFCSR)

Address offset: 0x113C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 07C0

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]			
												r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIV[11:0]												Res.	Res.	ASC	ESC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **RSN[3:0]**: Reference Slot Number

This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.

Bits 15:4 **SIV[11:0]**: Slot Interval Value

This field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 µs while the maximum value 4095 specifies the slot interval of 4096 µs. The default/reset value is 0x07C which corresponds to slot interval of 125 µs.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ASC**: Advance Slot Check

When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is

– equal to the reference slot number given in the RSN field

or

– ahead of the reference slot number by up to two slots

This bit is applicable only when the ESC bit is set.

Bit 0 **ESC**: Enable Slot Comparison

When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is

– equal to the reference slot number

or

– ahead of the reference slot number by one slot

When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.

Note: The UFO (UDP Fragmentation over IPv4)/TSO/USO should not be enabled along with TBS/AVB Slot number check. The UFO/TSO/USO involves multiple packets/segments/fragments transmission for single packet received from application and the slot number check are applicable for fetching of only first segment/fragment. As a result it might be difficult for the software to specify the slot number for subsequent packets.

Channel x current application transmit descriptor register (ETH_DMACxCATXDR)

Address offset: 0x1144 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURTDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTDESAPTR[31:0]**: Application Transmit Descriptor Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel 0 current application receive descriptor register (ETH_DMAC0CARXDR)

Address offset: 0x114C

Reset value: 0x0000 0000

The Channel 0 Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURRDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRDESAPTR[31:0]**: Application Receive Descriptor Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x current application transmit buffer register (ETH_DMACxCATXBR)

Address offset: 0x1154 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTBUFAPTR[31:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel 0 current application receive buffer register (ETH_DMAC0CARXBR)

Address offset: 0x115C

Reset value: 0x0000 0000

The Channel 0 Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRBUFAPTR[31:0]**: Application Receive Buffer Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x status register (ETH_DMACxSR)

Address offset: 0x1160 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]		
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rw	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **REB[2:0]**: Rx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AXI interface.

Bit [2]: Error during data transfer by Rx DMA when 1, no rror during data transfer by Rx DMA when 0.

Bit[1]: Error during descriptor access when 1, error during data buffer access when 0

Bit[0]: Error during read transfer when 1, error during write transfer when 0

This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16 **TEB[2:0]**: Tx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AXI interface.

Bit[2]: Error during data transfer by Tx DMA when 1, no error during data transfer by Tx DMA when 0

Bit[1]: Error during descriptor access when 1, error during data buffer access when 0

Bit[0]: Error during read transfer when 1, Error during write transfer when 0

This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15 **NIS**: Normal Interrupt Summary

Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACxIER register:

Bit 0: Transmit interrupt

Bit 2: Transmit buffer unavailable

Bit 6: Receive interrupt

Bit 11: Early receive interrupt

Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMACxIER register) affect the Normal Interrupt Summary bit.

This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14 **AIS**: Abnormal Interrupt Summary

Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACxIER register:

Bit 1: Transmit Process Stopped

Bit 7: Receive Buffer Unavailable

Bit 8: Receive Process Stopped

Bit 10: Early transmit interrupt

Bit 12: Fatal Bus Error

Bit 13: Context Descriptor Error

Only unmasked bits affect the Abnormal Interrupt Summary bit.

This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13 **CDE**: Context Descriptor Error

This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one‘s descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12 **FBE**: Fatal Bus Error

This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.

Bit 11 **ERI**: Early Receive Interrupt

This bit indicates that the DMA filled the first data buffer of the packet.The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early Transmit Interrupt

This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO.

Bit 9 **RWT**: Receive Watchdog Timeout

This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.

Bit 8 **RPS**: Receive Process Stopped

This bit is asserted when the Rx process enters the Stopped state.

Bit 7 **RBU**: Receive Buffer Unavailable

This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.

Bit 6 **RI**: Receive Interrupt

This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

The reception remains in the Running state.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit Buffer Unavailable

This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the *Debug status register (ETH_DMADSR)* register explains the Transmit Process state transitions.

To resume processing the Transmit descriptors, the application should do the following:

1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
2. Issue a Transmit Poll Demand command.

For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel.

Bit 1 **TPS**: Transmit Process Stopped

This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit Interrupt

This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

Channel missed frame count register (ETH_DMACMFCR)

Address offset: 0x116C

Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to bus error or due to programing RPF field in *Channel 0 receive control register (ETH_DMAC0RXCR)* register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
rc_r					rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter

When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped Packet Counters

This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in *Channel 0 receive control register (ETH_DMAC0RXCR)*. The counter gets cleared when this register is read.

Ethernet DMA register map and reset values

Table 537. ETH_DMA common register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	ETH_DMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]		Res.	Res.	Res.	Res.	TXPR	Res.	Res.	DSPW	Res.	Res.	Res.	TAA[2:0]			Res.	SWR
	Reset value															0	0					0			0				0	0	0		0
0x1004	ETH_DMASBMR	EN_LPI	LPI_XIT_PKT	Res.	Res.	Res.	Res.	WR_OSR_LMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]		Res.	Res.	ONEKBBE	AAL	Res.	AALE	Res.	Res.	BLEN256	BLEN128	BLEN64	BLEN32	BLEN16	BLEN8	BLEN4	FB
	Reset value	0	0					0	1							0	1			0	0		0			0	0	0	0	0	0	0	0
0x1008	ETH_DMAISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC1IS	DC0IS
	Reset value															0	0															0	0
0x100C	ETH_DMADSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS1[3:0]				RPS1[3:0]				TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	AXRHSTS	AXWHSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0
0x1010 - 0x101C	Reserved																																
0x1020	ETH_DMAA4TXACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THC[3:0]				Res.	Res.	Res.	Res.	TEC[3:0]				Res.	Res.	Res.	Res.	TDRC[3:0]			
	Reset value													0	0	0	0					0	0	0	0					0	0	0	0
0x1024	ETH_DMAA4RXACR	Res.	Res.	Res.	Res.	RDC[3:0]				Res.	Res.	Res.	Res.	RHC[3:0]				Res.	Res.	Res.	Res.	RPC[3:0]				Res.	Res.	Res.	Res.	RDWC[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x1028	ETH_DMAA4DACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRC[3:0]				Res.	Res.	TDWD[1:0]		TDWC[3:0]			
	Reset value																					0	0	0	0			0	0	0	0	0	0
0x102C - 0x103C	Reserved																																
0x1040	ETH_DMALPIEI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIEI[3:0]			
	Reset value																													0	0	0	0

Table 538. ETH_DMA_CH0 register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1100	ETH_DMAC0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1104	ETH_ DMAC0TXCR	Res.	Res.		Res.	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
	Reset value					0	0	0	0			0	0	0	0	0	0	0			0								0	0	0	0	0
0x1108	ETH_ DMAC0RXCR	RPF	Res.	Res.	Res.	RQOS[3:0]				Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]											Res.	Res.	Res.	SR
	Reset value	0				0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x110C - 0x1110	Reserved																																
0x1114	ETH_ DMAC0TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1118	Reserved																																
0x111C	ETH_ DMAC0RXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1120	ETH_ DMAC0TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1124	Reserved																																
0x1128	ETH_ DMAC0RXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x112C	ETH_ DMAC0TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1130	ETH_ DMAC0RXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0	0
0x1134	ETH_DMAC0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1138	ETH_ DMAC0RXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0

Table 538. ETH_DMA_CH0 register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x113C	ETH_DMAC0SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0			0	0
0x1140	Reserved																																
0x1144	ETH_DMAC0CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1148	Reserved																																
0x0114C	ETH_DMAC0CARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1150	Reserved																																
0x1154	ETH_DMAC0CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1158	Reserved																																
0x115C	ETH_DMAC0CARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1160	ETH_DMAC0SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1164 - 0x1168	Reserved																																
0x116C	ETH_DMACMFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Table 539. ETH_DMA_CH1 register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1180	ETH_DMAC1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1184	ETH_DMAC1TXCR	Res.	Res.		Res.	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
	Reset value					0	0	0	0			0	0	0	0	0	0	0			0								0	0	0	0	0
0x1188 - 0x1190	Reserved																																

Table 539. ETH_DMA_CH1 register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1194	ETH_DMAC1TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1198 - 0x119C	Reserved																																
0x11A0	ETH_DMAC1TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11A4 - 0x11A8	Reserved																																
0x11AC	ETH_DMAC1TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x11B0	Reserved																																
0x11B4	ETH_DMAC1IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x11B8	Reserved																																
0x11BC	ETH_DMAC1SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0
0x11C0	Reserved																																
0x11C4	ETH_DMAC1CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C8 - 0x11D0	Reserved																																
0x11D4	ETH_DMAC1CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
011D8 - 0x11DC	Reserved																																
0x11E0	ETH_DMAC1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x11E4- 0x11E8	Reserved																																

Refer to *Section 2.5 on page 153* for the register boundary addresses.

61.11.3 Ethernet MTL registers

Operating mode register (ETH_MTLOMR)

Address offset: 0x0C00

Reset value: 0x0000 0000

The Operating Mode register establishes the Transmit and Receive operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CNT CLR	CNT PRST	Res.	SCHALG[1:0]		Res.	Res.	RAA	DTX STS	Res.
						rw	rw		rw	rw			rw	rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CNTCLR**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence.

Bit 8 **CNTPRST**: Counters Preset

When this bit is set:

– *Tx queue x underflow register (ETH_MTLTXQxUR)* is initialized/preset to 0x7F0.

– Missed Packet and Overflow Packet counters in *Rx queue x missed packet and overflow counter register (ETH_MTLRXQxMPOCR)* is initialized/preset to 0x7F0

This bit is cleared automatically.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **SCHALG[1:0]**: Tx Scheduling Algorithm

This field indicates the algorithm for Tx scheduling:

00: Weighted round robin (WRR) algorithm

01 to 10: Reserved, must not be used.

11: Strict priority (SP) algorithm.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RAA**: Receive Arbitration Algorithm

This field is used to select the arbitration algorithm for the Rx side.

0: Strict priority (SP). Queue 0 has the lowest priority and the last queue has the highest priority.

1: Weighted Strict priority (WSP)

Bit 1 **DTXSTS**: Drop Transmit Status

When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

Bit 0 Reserved, must be kept at reset value.

Interrupt status register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q1IS	Q0IS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **Q1IS**: Queue 1 interrupt status

This bit indicates that an interrupt has been generated by Queue 1. To reset this bit, read ETH_MTLQ1ICSR register to identify the interrupt cause and clear the source.

Bit 0 **Q0IS**: Queue 0 interrupt status

This bit indicates that an interrupt has been generated by Queue 0. To reset this bit, read ETH_MTLQ0ICSR register to identify the interrupt cause and clear the source.

Tx queue x operating mode register (ETH_MTLTXQxOMR)

Address offset: 0x0D00 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Transmit Operating Mode register establishes the Transmit queue operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]			
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
									rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TQS[3:0]**: Transmit queue size

This field indicates the size of the allocated transmit queues in blocks of 256 bytes. Queue size range from 256 bytes (TQS = 0b0000) to 4096 bytes (TQS = 0b1111).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **TTC[2:0]**: Transmit Threshold Control

These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.

000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit Queue Enable

This field is used to enable/disable the transmit queue 0.

00: Not enabled
01: Enable in AV mode
10: Enabled
Others: Reserved, must not be used.

Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.

Bit 1 **TSF**: Transmit Store and Forward

When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush Transmit Queue

When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQ1OMR register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_tx_clk) should be active.

Tx queue x underflow register (ETH_MTLTXQxUR)

Address offset: 0x0D04 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **UFCNTOVF**: Overflow Bit for Underflow Packet Counter

This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

Bits 10:0 **UFFRMCNT[10:0]**: Underflow Packet Counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

Tx queue x debug register (ETH_MTLTXQxDR)

Address offset: 0x0D08 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]		
									r	r	r		r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTS FSTS	TXQ STS	TWC STS	TRCSTS[1:0]		TXQPA USED
										r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **STXSTSF[2:0]**: Number of Status Words in Tx Status FIFO of Queue

This field indicates the current number of status in the Tx Status FIFO of this queue.
When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PTXQ[2:0]**: Number of Packets in the Transmit Queue

This field indicates the current number of packets in the Tx queue.
When the DTXSTS bit of *Operating mode register (ETH_MTLOMR)* register is set to 1, this field does not reflect the number of packets in the Transmit queue.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TXSTSFSTS**: MTL Tx Status FIFO Full Status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

Bit 4 **TXQSTS**: MTL Tx Queue Not Empty Status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.

Bit 3 **TWCSTS**: MTL Tx Queue Write Controller Status

When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx Queue Read Controller Status

This field indicates the state of the Tx Queue Read Controller:

00: Idle state

01: Read state (transferring data to the MAC transmitter)

10: Waiting for pending Tx Status from the MAC transmitter

11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit Queue in Pause

When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:

- Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
- Reception of 802.3x Pause packet when PFC is disabled

Tx queue x ETS status register (ETH_MTLTXQxESR)

Address offset: 0x0D14 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x ETS Status register provides the average traffic transmitted in Queue x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:16]							
								r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ABS[23:0]**: Average Bits per Slot

This field contains the average transmitted bits per slot.

Queue x interrupt control status register (ETH_MTLQxICSR)

Address offset: 0x0D2C + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue x interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS
							rw								rc_w1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
						rw	rw							rw	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive Queue Overflow Interrupt Enable

When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RXOVFIS**: Receive Queue Overflow Interrupt Status

This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ABPSIE**: Average Bits Per Slot Interrupt Enable

When this bit is set, the MAC asserts the eth_sbd_intr_it interrupt when the average bits per slot status is updated.

When this bit is cleared, the interrupt is not asserted for such an event.

Bit 8 **TXUIE**: Transmit Queue Underflow Interrupt Enable

When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ABPSIS**: Average Bits Per Slot Interrupt Status

When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit.

Note: This bit is automatically set after 10 ms in 1000 Mbps mode (100 ms in 100 Mbps), once the reset is released, regardless of the Ethernet controller configuration.

Bit 0 **TXUNFIS**: Transmit Queue Underflow Interrupt Status

This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.

Rx queue x operating mode register (ETH_MTLRXQxOMR)

Address offset: 0x0D30 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Receive operating Mode register establishes the Receive queue operating modes and command.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD [2]
								rw	rw	rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFD[1:0]		Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_ TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
rw	rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **RQS[3:0]**: Receive Queue Size

This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The size of the queues is (RQS + 1) × 256 bytes.

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **RFD[2:0]**: Threshold for Deactivating Flow Control (in Half-duplex and Full-duplex modes)

These bits control the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation:

0: Full minus 1 Kbyte

1: Full minus 1.5 Kbyte

2: Full minus 2 Kbytes

3: Full minus 2.5 Kbytes

4: Full minus 3 Kbytes

5: Full minus 3.5 Kbytes

Other values: reserved

The deassertion is effective only after flow control is asserted.

Note: The value must be programmed in such a way to make sure that the threshold is a positive number.

To have the flow control activated for a fill level of 3 Kbytes and deactivated when the fill level is 1 Kbyte, program:

RFA[2:0] = 0 (full – 1 Kbyte)

RFD[2:0] = 4 (full – 3 Kbytes)

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **RFA[2:0]**: Threshold for Activating Flow Control (in Half-duplex and Full-duplex)

These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, see RFD[2:0].

Bit 7 **EHFC**: Enable Hardware Flow Control

When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.

Bit 6 **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets

When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.

When this bit is reset, all error packets are dropped if the FEP bit is reset.

Bit 5 **RSF**: Receive Queue Store and Forward

When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Bit 4 **FEP**: Forward Error Packets

When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.

When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.

Bit 3 **FUP**: Forward Undersized Good Packets

When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control

These bits control the threshold level of the MTL Rx queue (in bytes):

00: 64

01: 32

10: 96

11: 128

The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.

This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

Rx queue x missed packet and overflow counter register (ETH_MTLRXQxMPOCR)

Address offset: 0x0D34 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed Packet Counter Overflow Bit

When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed Packet Counter

This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.

This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit

When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow Packet Counter

This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.

Rx queue x debug register (ETH_MTLRXQxDR)

Address offset: 0x0D38 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Receive Debug register gives the debug status of various blocks related to the Receive queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PRXQ[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
										r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PRXQ[13:0]**: Number of Packets in Receive Queue
This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256 Kbyte/16 bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **RXQSTS[1:0]**: MTL Rx Queue Fill-Level Status
This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **RRCSTS[1:0]**: MTL Rx Queue Read Controller State
This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0 **RWCSTS**: MTL Rx Queue Write Controller Active Status
When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.

Rx queue x control register (ETH_MTLRXQxCR)

Address offset: 0x0D3C + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RXQ_FRM_ARBIT**: Receive Queue Packet Arbitration

When this bit is set, the Ethernet peripheral drives the packet data to the DMA such that the entire packet data of currently-selected queue is transmitted before switching to other queue.

When this bit is reset, the Ethernet peripheral drives the packet data to the DMA such that the following amount of data of currently-selected queue is transmitted before switching to other queue:

– PBL amount of data

or

– complete data of a packet

The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet peripheral drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).

Bits 2:0 **RXQ_WEGT[2:0]**: Receive Queue Weight

This field indicates the weight assigned to the Rx queue x. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle.

This field needs to be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on.

Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the Rx queue x control register (ETH_MTLRXQxCR) must be programmed before the Rx queue x operating mode register (ETH_MTLRXQxOMR).

Tx queue 1 ETS control register (ETH_MTLTXQ1ECR)

Address offset: 0x0D50

Reset value: 0x0000 0000

The Queue ETS Control register controls the enhanced transmission selection operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
									rw	rw	rw	rw	rw		

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SLC[2:0]**: Slot Count

If the credit-based shaper algorithm is enabled, the software can program the number of slots (of 125 us duration) over which the average transmitted bits per slot, provided in *Tx queue x ETS status register (ETH_MTLTXQxESR)*, need to be computed for Queue 1. The encoding is as follows:

000: 1 Slot
001: 2 Slots
010: 4 Slots
011: 8 Slots
100: 16 Slots
101 to 111: Reserved, must not be used.

Bit 3 **CC**: Credit Control

When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel 1. The credit accumulates even when there is no packet waiting in Channel 1 and another channel is transmitting.

When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel 1. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated.

Bit 2 **AVALG**: AV Algorithm

When Queue 1 is programmed for AV, this field configures the scheduling algorithm for this queue:

This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected.

Bits 1:0 Reserved, must be kept at reset value.

Tx queue x quantum weight register (ETH_MTLTXQxQWR)

Address offset: 0x0D18 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

This register contains the weights for the Weighted Round Robin (WRR) or IdleSlopeCredit for the queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	ISCQW[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **ISCQW[13:0]**: IdleSlopeCredit or Weights

idleSlopeCredit

When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode.

Weights

If WRR algorithm is enabled for queue x generic traffic (not AVB), this field contains the weight for queue x. The maximum value is 0x64. Bits [13:7] must be written to zero.

Note: In multiple Queue configuration this field must be programmed to some non-zero value when multiple queues are enabled, or a single queue other than Queue 0 is enabled. This field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both Receive and Transmit. In Receive, the register is Operating mode register (ETH_MTLOMR).

The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue w.r.t. total BW or time.

Tx queue 1 send slope credit register (ETH_MTLTXQ1SSCR)

Address offset: 0x0D5C

Reset value: 0x0000 0000

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	SSC[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **SSC[13:0]**: sendSlopeCredit Value

When AV operation is enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns and 8 ns for 100 Mbps and 1000 Mbps respectively) when the credit decreases. The software should program this field with computed credit in bits per cycle scaled by 1 024. The maximum value is portTransmitRate, that is 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when queue 1 is selected for transmission.

Tx Queue 1 hiCredit register (ETH_MTLTXQ1HCR)

Address offset: 0x0D60

Reset value: 0x0000 0000

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HC[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **HC[28:0]**: hiCredit Value

This field contains the hiCredit value required for the credit-based shaper algorithm when the AV feature is enabled. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1 024.

The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16 384 bytes or 131 072 bits). The value to be specified is 131 072 ×1 024 = 134 217 728 or 0x0800 0000.

Tx queue 1 loCredit register (ETH_MTLTXQ1LCR)

Address offset: 0x0D64

Reset value: 0x0000 0000

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LC[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **LC[28:0]**: loCredit Value

When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1 024. The maximum value is maxFrameSize transmitted from this queue, that is, 16 384 × 8 × 1 024 = 134 217 728 or 0x0800 0000. Because it is a negative value, the programmed value is 2's complement of the value, that is, 0x1800 0000.

Ethernet MTL register map and reset values

Table 540. ETH_MTL register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C00	ETH_MTLOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	SCHALG[1:0]		Res.	Res.	RAA	DTXSTS	Res.
	Reset value																							0	0		0	0			0	0	
0x0C04 - 0x0C1C	Reserved																																
0x0C20	ETH_MTLISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q1IS	Q0IS
	Reset value																															0	0
0x0C24 - 0x0CFC	Reserved																																
0x0C40	Reserved																																
0x0D00	ETH_ MTLTXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value													0	0	0	0										0	0	0	0	0	0	0
0x0D04	ETH_ MTLTXQ0UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D08	ETH_ MTLTXQ0DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D0C - 0x0D10	Reserved																																
0x0D14	ETH_ MTLTXQ0ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D18	ETH_ MTLTXQ0QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x0D28	Reserved																																

Table 540. ETH_MTL register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D2C	ETH_MTLQ0ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0							0	0
0x0D30	ETH_ MTLRXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD[2:0]			Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value									0	0	0	0				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D34	ETH_ MTLRXQ0MPO CR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D38	ETH_ MTLRXQ0DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0
0x0D3C	ETH_ MTLRXQ0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBI	RXQ_WEGT[2:0]		
	Reset value																													0	0	0	0
0x0D40	ETH_ MTLTXQ1OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value													0	0	0	0										0	0	0	0	0	0	0
0x0D44	ETH_ MTLTXQ1UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D48	ETH_ MTLTXQ1DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D4C	Reserved																																

Table 540. ETH_MTL register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D50	ETH_ MTLTXQ1ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
	Reset value																										0	0	0	0	0		
0x0D54	ETH_ MTLTXQ1ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D58	ETH_ MTLTXQ1QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D5C	ETH_ MTLTXQ1SSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D60	ETH_ MTLTXQ1HCR	Res.	Res.	Res.	HC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D64	ETH_ MTLTXQ1LCR	Res.	Res.	Res.	LC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D68	Reserved																																
0x0D6C	ETH_MTLQ1ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0							0	0
0x0D70	ETH_ MTLRXQ1OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD[2:0]			Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value									0	0	0	0				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D74	ETH_MTLRXQ1M POCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D78	ETH_ MTLRXQ1DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0

Table 540. ETH_MTL register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D7C	ETH_MTLRXQ1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBI	RXQ_WEGT[2:0]		
	Reset value																													0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

61.11.4 Ethernet MAC and MMC registers

Operating mode configuration register (ETH_MACCR)

Address offset: 0x0000

Reset value: 0x0000 0000

The MAC Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **ARPEN**: ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

Bits 30:28 **SARC[2:0]**: Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

010:the MAC inserts the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

011: the MAC replaces the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

110: the MAC inserts the content of the MAC Address 1 registers (*MAC Address x high register (ETH_MACAxHR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets

111: the MAC replaces the content of the MAC Address 1 registers (*MAC Address x high register (ETH_MACAxHR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

Others: Reserved, must not be used.

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bit 27 **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.

000: 96 bit times

001: 88 bit times

010: 80 bit times

...

111: 40 bit times

This range of minimum IPG is valid in Full-duplex mode.

In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.

When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.

The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.

When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).

The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.

When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 541: Giant Packet Status based on S2KP and JE Bits*.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 542: Packet Length based on the CST and ACS bits.

Bit 20 **ACS**: Automatic Pad or CRC Stripping

When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.

When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

Note: For information about how the settings of CST bit and this bit impact the packet length, see Table 542: Packet Length based on the CST and ACS bits.

Bit 19 **WD**: Watchdog Disable

When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.

When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18 **BE**: Packet Burst Enable

When this bit is set, the MAC allows packet bursting during transmission in the GMII Half-duplex mode.

Bit 17 **JD**: Jabber Disable

When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.

When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16 **JE**: Jumbo Packet Enable

When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.

For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 541: Giant Packet Status based on S2KP and JE Bits*.

Bit 15 **PS**: Port Select

This bit selects the Ethernet line speed.

0: For 1000 Mbps operations

1: For 10 or 100 Mbps operations

In 10 or 100 Mbps operations, this bit, along with Bit FES, selects the exact line speed.

Bit 14 **FES**: MAC Speed

This bit selects the speed in the 10/100 Mbps mode:

0: 10 Mbps

1: 100 Mbps

Bit 13 **DM**: Duplex Mode

When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.

Bit 12 **LM**: Loopback Mode

When this bit is set, the MAC operates in the loopback mode at GMII or MII. The GMII Rx clock input (eth_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode

When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.

When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10 **DO**: Disable Receive Own

When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.

This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.

Bit 9 **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the GMII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8 **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:

00: k= min (n, 10)

01: k = min (n, 8)

10: k = min (n, 4)

11: k = min (n, 1)

where n = retransmission attempt

The random integer r takes the value in the range 0 <= r < 2^k.

This bit is applicable only in the Half-duplex mode.

Bit 4 **DC**: Deferral Check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode.

If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII.

The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the Half-duplex mode.

Bits 3:2 **PRELEN[1:0]**: Preamble Length for Transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.

00: 7 bytes of preamble

01: 5 bytes of preamble

10: 3 bytes of preamble

11: Reserved, must not be used

Bit 1 **TE**: Transmitter Enable

When this bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver Enable

When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface.

Table 541 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 541. Giant Packet Status based on S2KP and JE Bits(1)

Length/Type Field	Received Packet Length	S2KP	JE	Giant Packet Status
Untagged packet	> 1,518	0	0	1
	> 2,000	1	0	1
	> 9,018	x	1	1
VLAN tagged packet	> 1,522	0	0	1
	> 2,000	1	0	1
	> 9,022	x	1	1

1. For all other combinations, the Giant Packet status is 0.

Table 542 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 542. Packet Length based on the CST and ACS bits

Received Packet Length	CST	ACS	FCS Stripping Done
< 1,536	x	0	No
	x	1	Yes (for Ethernet packets)
≥ 1,536	0	x	No
	1	x	Yes (for Type packets)

Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004

Reset value: 0x0000 0000

The MAC Extended Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC
		rw	rw	rw	rw	rw	rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GPSL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in *Operating mode configuration register (ETH_MACCR)*, gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:

- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24 **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in *Operating mode configuration register (ETH_MACCR)* together as minimum IPG greater than 96 bit times in steps of 8 bit times.

When this bit is reset, the MAC ignores EIPG field and interprets IPG field in *Operating mode configuration register (ETH_MACCR)* as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the *MAC Address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC Address x low register (ETH_MACAxLR)*. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).

When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.

Bit 17 **SPEN**: Slow Protocol Detection Enable

When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes.

When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets

When this bit is set, the MAC receiver does not check the CRC field in the received packets.

When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit

If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.

For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008

Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE
rw										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RA**: Receive All

When this bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.

When this bit is reset, the receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DNTU**: Drop Non-TCP/UDP over IP Packets

When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.

Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable

When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.

When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable

When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter

When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.

When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9 **SAF**: Source Address Filter Enable

When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.

When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

Note: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering

When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.

When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets

These bits control the forwarding of all control packets (including unicast and multicast Pause packets).

00: The MAC filters all control packets from reaching the application.

01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.

10: The MAC forwards all control packets to the application even if they fail the Address filter.

11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets

When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.

When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast

When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.

Bit 3 **DAIF**: DA Inverse Filtering

When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast

When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.

When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast

When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.

When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode

When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

Watchdog timeout register (ETH_MACWTR)

Address offset: 0x000C

Reset value: 0x0000 0000

The Watchdog Timeout register controls the watchdog timeout for received packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PWE**: Programmable Watchdog Enable

When this bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in *Operating mode configuration register (ETH_MACCR)* register.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WTO[3:0]**: Watchdog Timeout

When the PWE bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes

0x1: 3 Kbytes

0x2: 4 Kbytes

0x3: 5 Kbytes

..

0xC: 14 Kbytes

0xD: 15 Kbytes

0xE: 16383 Bytes

0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010

Reset value: 0x0000 0000

The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT31T0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT31T0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT31T0[31:0]**: MAC Hash Table First 32 Bits
This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014

Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 6‘b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT63T32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT63T32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT63T32[31:0]**: MAC Hash Table Second 32 Bits
This field contains the second 32 Bits [63:32] of the Hash table.

VLAN tag register (ETH_MACVTR)

Address offset: 0x0050

Reset value: 0x0000 0000

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EIVLRXS**: Enable Inner VLAN Tag in Rx Status

When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **EIVLS[1:0]**: Enable Inner VLAN Tag Stripping on Receive

This field indicates the stripping operation on inner VLAN Tag in received packet:

00: Do not strip

01: Strip if VLAN filter passes

10: Strip if VLAN filter fails

11: Always strip

Bit 27 **ERIVLT**: Enable Inner VLAN Tag

When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.

Bit 26 **EDVLP**: Enable Double VLAN Processing

When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).

Bit 25 **VTHM**: VLAN Tag Hash Table Match Enable

When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table.

When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison.

When this bit is reset, the VLAN Hash Match operation is not performed.

Bit 24 **EVLRXS**: Enable VLAN Tag in Rx status

When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive
This field indicates the stripping operation on the outer VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 20 **DOVLTC**: Disable VLAN Type Check
When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19 **ERSVLM**: Enable Receive S-VLAN Match
When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18 **ESVL**: Enable S-VLAN
When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17 **VTIM**: VLAN Tag Inverse Match Enable
When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16 **ETV**: Enable 12-Bit VLAN Tag Comparison
When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:0 **VL[15:0]**: VLAN Tag Identifier for Receive Packets
This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
When the ETV bit is set, only the VID is used for comparison.
If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058

Reset value: 0x0000 0000

When the VTHM bit of *VLAN tag register (ETH_MACVTR)* register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of *VLAN tag register (ETH_MACVTR)* register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLHT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VLHT[15:0]**: VLAN Hash Table
This field contains the 16-bit VLAN Hash Table.

VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]	
											rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.

10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets
This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Tx Queue 0 flow control register (ETH_MACQ0TXFCR)

Address offset: 0x0070

Reset value: 0x0000 0000

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
								rw	rw	rw	rw			rw	rw

Bits 31:16 **PT[15:0]**: Pause Time
This field holds the value to be used in the Pause Time field in the Tx control packet. I

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DZPQ**: Disable Zero-Quanta Pause
When this bit is set, it disables the automatic generation of the zero-quanta Pause packets.
When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Bits 6:4 **PLT[2:0]**: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)

001: Pause Time minus 28 Slot Times (PT -28 slot times)

010: Pause Time minus 36 Slot Times (PT -36 slot times)

011: Pause Time minus 144 Slot Times (PT -144 slot times)

100: Pause Time minus 256 Slot Times (PT -256 slot times)

101: Pause Time minus 512 Slot Times (PT -512 slot times)

110 to 111: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface.

This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TFE**: Transmit Flow Control Enable

Full-duplex mode: when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

Half-duplex mode: when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 **FCB_BPA**: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

Full-Duplex mode: this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

Half-duplex mode: When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.

Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090

Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in *MAC Address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC Address x low register (ETH_MACAxLR)*.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01 80 C2 00 00 01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0 **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.

Rx Queue control register (ETH_MACRXQCR)

Address offset: 0x0094

Reset value: 0x0000 0000

The Receive Queue Control register controls the routing of unicast and multicast packets that fail the Destination or Source address filter to the Rx queues

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VFFQ	VFFQE
														rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	MFFQ	MFFQE	Res.	Res.	Res.	Res.	Res.	Res.	UFFQ	UFFQE
						rw	rw							rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **VFFQ**: VLAN Tag Filter Fail Packets Queue

This field holds the Rx queue number to which the tagged packets failing the Destination or Source Address filter (and UFFQE/MFFQE not enabled) or failing the VLAN tag filter must be routed to. This field is valid only when the VFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 16 **VFFQE**: VLAN Tag Filter Fail Packets Queuing Enable

When this bit is set, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter are routed to the Rx Queue Number programmed in the VFFQ. When this bit is reset, the tagged packets which fail the Destination or Source Address filter or fail the VLAN tag filter are routed based on other routing options. This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

0: Disabled

1: Enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MFFQ**: Multicast Address Filter Fail Packets Queue.

This field holds the Rx queue number to which the Multicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the MFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 8 **MFFQE**: Multicast Address Filter Fail Packets Queuing Enable.

When this bit is set, the Multicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the MFFQ.

When this bit is reset, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options.

This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **UFFQ**: Unicast Address Filter Fail Packets Queue.

This field holds the Rx queue number to which the Unicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the UFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 0 **UFFQE**: Unicast Address Filter Fail Packets Queuing Enable.

When this bit is set, the Unicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the UFFQ.

When this bit is reset, the Unicast packets which fail the Destination or Source Address filter is routed based on other routing options.

This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

Rx queue control 0 register (ETH_MACRXQC0R)

Address offset: 0x00A0

Reset value: 0x0000 0000

The Receive Queue Control 0 register controls the queue management in the MAC receiver.

Note: In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ1EN[1:0]		RXQ0EN[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **RXQ1EN[1:0]**: Receive Queue 1 Enable

This field indicates whether Rx queue 1 is enabled for AV or Generic traffic.

00: Not enabled

01: Queue 1 enabled for AV

10: Queue 1 enabled for Generic traffic

11: Reserved, must not be used

Bits 1:0 **RXQ0EN[1:0]**: Receive Queue 0 Enable

This field indicates whether Rx queue 0 is enabled for AV or Generic traffic.

00: Not enabled

01: Queue 0 enabled for AV

10: Queue 0 enabled for Generic traffic

11: Reserved, must not be used

Rx queue control 1 register (ETH_MACRXQC1R)

Address offset: 0x00A4

Reset value: 0x0000 0000

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV and untagged packets to the Rx queues.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPQC[1:0]		TACPQE	MCBCQEN	Res.	MCBCQ[2:0]		
								rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UPQ[2:0]			Res.	Res.	Res.	Res.	Res.	PTPQ[2:0]			Res.	AVCPQ 2	AVCPQ 1	AVCPQ 0
	rw	rw	rw						rw	rw	rw		rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **TPQC[1:0]**: Tagged PTP over Ethernet Packets Queuing Control
This field controls the routing of the VLAN tagged PTPoE packets.
00: VLAN tagged PTPoE packets are routed as generic VLAN tagged packet (based on PSRQ for only non-AV enabled Rx queues).
01: VLAN tagged PTPoE packets are routed to Rx Queue specified by PTPQ field (this Rx Queue can be enabled for AV or non-AV traffic)
10: VLAN tagged PTPoE packets are routed to only AV enabled Rx queues based on PSRQ.
11: Reserved

Bit 21 **TACPQE**: Tagged AV Control Packets Queuing Enable
When set, the MAC routes the received Tagged AV control packets to the Rx queue specified by AVCPQ field.
When reset, the MAC routes the received Tagged AV control packets based on the tag priority matching the PSRQ fields in ETH_MACRXQC2R register.
This field is reserved if Rx queues are not selected for AV feature.

Bit 20 **MCBCQEN**: Multicast and Broadcast Queue Enable
This bit specifies that Multicast or Broadcast packets routing to the Rx queue is enabled and the Multicast or Broadcast packets must be routed to Rx queue specified in MCBCQ field.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **MCBCQ[2:0]**: Multicast and Broadcast Queue
This field specifies the Rx queue onto which Multicast or Broadcast packets are routed. Any Rx queue enabled for Generic/AV traffic can be used to route the Multicast or Broadcast packets.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **UPQ[2:0]**: Untagged Packet Queue

This field indicates the Rx queue to which Untagged Packets are to be routed. Any Rx queue enabled for Generic/AV traffic can be used to route the Untagged Packets.

000: Rx queue 0

001: Rx queue 1

Others: Reserved, must not be used

Bits 11:7 Reserved, must be kept at reset value.

Bits 6:4 **PTPQ[2:0]**: PTP Packets Queue

This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed.

000: Rx queue 0

001: Rx queue 1

Others: Reserved, must not be used

When the AV8021ASMEN bit of *Timestamp control Register (ETH_MACTSCR)* is set, only untagged PTP over Ethernet packets are routed on an Rx queue. If the bit is not set, then based on programming of TPQC field, both tagged and untagged PTPoE packets can be routed to this Rx Queue.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AVCPQ[2:0]**: AV Untagged Control Packets Queue

This field specifies the Receive queue on which the received AV untagged control packets are routed:

000: Receive Queue 0

001: Receive Queue 1

Others: Reserved, must not be used

The AV tagged control and data packets are routed based on PSRQ field in the Transmit Flow Control Register of corresponding queue.

This field is reserved if Receive queues are not selected for AV feature.

Rx queue control 2 register (ETH_MACRXQC2R)

Address offset: 0x00A8

Reset value: 0x0000 0000

This register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queue 0 and 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSRQ1[7:0]								PSRQ0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **PSRQ1[7:0]**: Priorities Selected in the Receive Queue 1

This field defines the priorities assigned to Rx queue 1. All packets with priorities that match the values set in this field are routed to Rx queue 1.

For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

Bits 7:0 **PSRQ0[7:0]**: Priorities Selected in the Receive Queue 0

This field defines the priorities assigned to Rx queue 0. All packets with priorities that match the values set in this field are routed to Rx queue 0.

For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

Interrupt status register (ETH_MACISR)

Address offset: 0x00B0

Reset value: 0x0000 0000

The Interrupt Status register contains the status of interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	RGSMIIIS
	rc_r	rc_r	rc_r		r	r	r			r	r	r			r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIS**: Receive Status Interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the *Rx Tx status register (ETH_MACRXTXSR)*. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 13 **TXSTSIS**: Transmit Status Interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the *Rx Tx status register (ETH_MACRXTXSR)*:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber Timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 12 **TSIS**: Timestamp Interrupt Status

If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

– The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.

– There is an overflow in the Seconds register.

– The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the *Timestamp status register (ETH_MACTSSR)*.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC Transmit Interrupt Status

This bit is set high when an interrupt is generated in the *MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive Interrupt Status

This bit is set high when an interrupt is generated in the *MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMCIS**: MMC Interrupt Status

This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIIS**: LPI Interrupt Status

This bit is set for any LPI state entry or exit in the MAC transmitter or receiver. This bit is cleared when the TLPIEN bit of *LPI control and status register (ETH_MACLCSR)* is read.

Bit 4 **PMTIS**: PMT Interrupt Status

This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the *PMT control status register (ETH_MACPCSR)*.

Bit 3 **PHYIS**: PHY Interrupt

This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RGSMIIIS**: RGMII Interrupt Status

This bit is set by any change in value of the Link Status of RGMII interface (LNKSTS bit in *PHYIF control status register (ETH_MACPHYCSR)* register). This bit is cleared when the ETH_MACPHYCSR register is read.

Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4

Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	RGSMIIIE
	rw	rw	rw							rw	rw	rw			rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIE**: Receive Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 13 **TXSTSIE**: Transmit Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 12 **TSIE**: Timestamp Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **LPIIE**: LPI Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 4 **PMTIE**: PMT Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 3 **PHYIE**: PHY Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RGSMIIIE**: RGMII Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of RGSMIIIS bit in *Interrupt status register (ETH_MACISR)*.

Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8

Reset value: 0x0000 0000

The Receive Transmit Status register contains the Receive and Transmit Error status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
							rc_r			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RWT**: Receive Watchdog Timeout

This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EXCOL**: Excessive Collisions

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after the first collision and the packet transmission is aborted.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **LCOL**: Late Collision

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode).

This bit is not valid if the Underflow error occurs.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 3 **EXDEF**: Excessive Deferral

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)* and the DC bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000 Mbps mode or when Jumbo packet is enabled).

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **NCARR**: No Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TJT**: Transmit Jabber Timeout

This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

PMT control status register (ETH_MACPCSR)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw			r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
					rw	rw			r	rc_r			rw	rw	rw

Bit 31 **RWKFILTRST**: Remote Wake-up Packet Filter Register Pointer Reset

When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24 **RWKPTR[4:0]**: Remote wake-up FIFO Pointer

This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **RWKPFE**: Remote wake-up Packet Forwarding Enable

When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.

The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.

Note: If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9 **GLBLUCAST**: Global Unicast

When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RWKPRCVD**: Remote wake-up Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5 **MGKPRCVD**: Magic Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RWKPKTEN**: Remote wake-up Packet Enable

When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1 **MGKPKTEN**: Magic Packet Enable

When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0 **PWRDWN**: Power Down

When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACRWKPFR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACRWKPFR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MACRWKPFR[31:0]**: Remote wake-up packet filter

Refer to *Table 509*, *Table 510* and *Table 511* for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	PLSEN	PLS	LPIEN
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
						r	r					r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **LPITCSE**: LPI Tx Clock Stop Enable

When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.

After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.

When LPITE is 0, the LPI Auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side.

If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxQxOMR, when the MAC is in the LPI mode, it exits the LPI mode.

When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 **PLSEN**: PHY Link Status Enable

This bit enables the link status received on the RGMII Receive paths to be used for activating the LPI LS TIMER.

When this bit is set, the MAC uses the link-status bits of the ETH_MACPHYCSR register and the PLS bit for the LPI LS Timer trigger. When this bit is reset, the MAC ignores the link-status bits of the ETH_MACPHYCSR register and takes only the PLS bit.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.

When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.

This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2 **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1 **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

LPI timers control register (ETH_MACLTCR)

Address offset: 0x00D4

Reset value: 0x03E8 0000

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TWT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **LST[9:0]**: LPI LS Timer
This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.

Bits 15:0 **TWT[15:0]**: LPI TW Timer
This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8

Reset value: 0x0000 0000

This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of *LPI control and status register (ETH_MACLCSR)* register is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:16]			
												rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LPIET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **LPIET[19:0]**: LPI Entry Timer
This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.
Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

One-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC

Reset value: 0x0000 0063

This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 TIC_1US_CNTR[11:0]: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs (subtract 1 from the value before programming).

For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).

This is required to generate the 1 µs events that are used to update some of the EEE related counters.

PHYIF control status register (ETH_MACPHYCSR)

Address offset: 0x00F8

Reset value: 0x0000 0000

The PHY Interface Control and Status register indicates the status signals received by the RGMII interface from the PHY.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNKSTS	LNKSPEED[1:0]		LNKMOD
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUD	TC
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **LNKSTS**: Link Status

This bit indicates whether the link is up or down.

0: Link down

1: Link up

Bits 18:17 **LNKSPEED[1:0]**: Link Speed

This bit indicates the current speed of the link:

00: 2.5 MHz

01: 25 MHz

10: 125 MHz

11: Reserved, must not be used

Bit 16 **LNKMOD**: Link Mode

This bit indicates the current operating mode of the link:

0: Half-duplex mode

1: Full-duplex mode

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LUD**: Link Up or Down

This bit indicates whether the link is up or down during transmission of configuration in the RGMII interface.

0: Link Down

1: Link Up

Bit 0 **TC**: Transmit Configuration in RGMII

When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII port. When this bit is reset, no such information is driven to the PHY (see *Section 61.6.4: Reduced media independent interface (RMII)*).

Version register (ETH_MACVR)

Address offset: 0x0110

Reset value: 0x0000 4042

The version register identifies the version of the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USERVER[7:0]								SNPSVER[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **USERVER[7:0]**: ST-defined version

Bits 7:0 **SNPSVER[7:0]**: IP version

Debug register (ETH_MACDR)

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS
													r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **TFCSTS[1:0]**: MAC Transmit Packet Controller Status

This field indicates the state of the MAC Transmit Packet Controller module:

00: Idle state

01: Waiting for one of the following:

– Status of the previous packet

– IPG or backoff period to be over

10: Generating and transmitting a Pause control packet (in Full-duplex mode)

11: Transferring input packet for transmission

Bit 16 **TPESTS**: MAC GMII or MII Transmit Protocol Engine Status

When this bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **RFCFCSTS[1:0]**: MAC Receive Packet Controller FIFO Status

When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.

Bit 0 **RPESTS**: MAC GMII or MII Receive Protocol Engine Status

When this bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state.

HW feature 0 register (ETH_MACHWF0R)

Address offset: 0x011C

Reset value: 0x0A0D 73F7

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]		MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	r	r	r			r	r	r	r	r	r	r	r	r	r

Bits 31:28 **ACTPHYSEL[3:0]**: Active PHY selected
When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
0x0: GMII or MII
0x1: RGMII
0x2: SGMII
0x3: TBI
0x4: RMII
0x5: RTBI
0x6: SMII
Others: Reserved, must not be used

Bit 27 **SAVLANINS**: Source Address or VLAN Insertion Enable
This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25 **TSSTSSEL[1:0]**: Timestamp System Time Source
This bit indicates the source of the Timestamp system time:
01: Internal
10: External
11: Both
00: Reserved, must not be used
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24 **MACADR64SEL**: MAC Addresses 64-127 Selected
This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23 **MACADR32SEL**: MAC Addresses 32-63 Selected
This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18 **ADDMACADRSEL[4:0]**: MAC Addresses 1-31 Selected
This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RXCOESEL**: Receive Checksum Offload Enabled
This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXCOESEL**: Transmit Checksum Offload Enabled
This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected

Bit 13 **EEESEL**: Energy Efficient Ethernet Enabled
This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12 **TSSEL**: IEEE 1588-2008 Timestamp Enabled
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ARPOFFSEL**: ARP Offload Enabled
This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8 **MMCSEL**: RMON Module Enable
This bit is set to 1 when the Enable MAC management counters (MMC) option is selected

Bit 7 **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6 **RWKSEL**: PMT Remote Wake-up Packet Enable
This bit is set to 1 when the Enable Remote Wake-up Packet Detection option is selected

Bit 5 **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4 **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3 **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2 **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1 **GMIISEL**: 1000 Mbps Support
This bit is set to 1 when 1000 Mbps is selected as operating mode.

Bit 0 **MIISEL**: 10 or 100 Mbps Support
This bit is set to 1 when 10/100 Mbps is selected as operating mode.

HW feature 1 register (ETH_MACHWF1R)

Address offset: 0x0120

Reset value: 0x1114 1945

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
	r	r	r	r		r	r	r		r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					Res.	RXFIFOSIZE[4:0]				
r	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
This field indicates the total number of L3 or L4 filters:
0000: No L3 or L4 Filter
0001: 1 L3 or L4 Filter
0010: 2 L3 or L4 Filters
..
1000: 8 L3 or L4

Bit 26 Reserved, must be kept at reset value.

Bits 25:24 **HASHTBLSZ[1:0]**: Hash Table Size
This field indicates the size of the Hash table:
00: No Hash table
01: 64
10: 128
11: 256

Bit 23 **POUOST**: One Step for PTP over UDP/IP Feature Enable
This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **RAVSEL**: Rx Side Only AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20 **AVSEL**: AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19 **DBGMEMA**: DMA Debug Registers Enable
This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18 **TSOEN**: TCP Segmentation Offload Enable
This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17 **SPHEN**: Split Header Feature Enable
This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16 **DCBEN**: DCB Feature Enable
This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14 **ADDR64[1:0]**: Address width
This field indicates the configured address width.
00: 32 bits
Others: Reserved, must not be used

Bit 13 **ADVTHWORD**: IEEE 1588 High Word Register Enable
This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12 **PTOEN**: PTP Offload Enable
This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11 **OSTEN**: One-Step Timestamping Enable
This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.

Bits 10:6 **TXFIFOSIZE[4:0]**: MTL Transmit FIFO Size

This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011 to 11111: Reserved, must not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL Receive FIFO Size

This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011: 256 Kbytes
01100 to 11111: Reserved, must not be used

HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x4104 0041

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]	
	r	r	r		r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
r	r	r	r			r	r	r	r			r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **AUXSNAPNUM[2:0]**: Number of Auxiliary Snapshot Inputs
This field indicates the number of auxiliary snapshot inputs:
000: No auxiliary input
001: 1 auxiliary input
010: 2 auxiliary inputs
011: 3 auxiliary inputs
100: 4 auxiliary inputs
101 to 111: Reserved, must not be used

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PPSOUTNUM[2:0]**: Number of PPS Outputs
This field indicates the number of PPS outputs:
000: No PPS output
001: 1 PPS output
010: 2 PPS outputs
011: 3 PPS outputs
100: 4 PPS outputs
101 to 111: Reserved, must not be used

Bits 23:22 **TDCSZ[1:0]**: Tx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 21:18 **TXCHCNT[3:0]**: Number of DMA Transmit Channels
This field indicates the number of DMA Transmit channels:
0000: 1 DMA Tx Channel
0001: 2 DMA Tx Channels
..
0111: 8 DMA Tx

Bits 17:16 **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 15:12 **RXCHCNT[3:0]**: Number of DMA Receive Channels
This field indicates the number of DMA Receive channels:
0000: 1 DMA Rx Channel
0001: 2 DMA Rx Channels
..
0111: 8 DMA Rx

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **TXQCNT[3:0]**: Number of MTL Transmit Queues
This field indicates the number of MTL Transmit queues:
0000: 1 MTL Tx queue
0001: 2 MTL Tx queues
..
0111: 8 MTL Tx

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **RXQCNT[3:0]**: Number of MTL Receive Queues
This field indicates the number of MTL Receive queues:
0000: 1 MTL Rx queue
0001: 2 MTL Rx queues
..
0111: 8 MTL Rx

HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128

Reset value: 0x0000 0020

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
										r	r		r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4 **CBTISEL**: Queue/Channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx feature is selected.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **NRVF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
000: No Extended Rx VLAN Filters
001: 4 Extended Rx VLAN Filters
010: 8 Extended Rx VLAN Filters
011: 16 Extended Rx VLAN Filters
100: 24 Extended Rx VLAN Filters
101: 32 Extended Rx VLAN Filters
110 to 111: Reserved, must not be used

MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO Address register controls the management cycles to external PHY through a management interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]				
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	GB
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PSE**: Preamble Suppression Enable

When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

Bit 26 **BTB**: Back to Back transactions

When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (GMII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

Bits 25:21 **PA[4:0]**: Physical Layer Address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.
This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Bits 20:16 **RDA[4:0]**: Register/Device Address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **NTC[2:0]**: Number of Training Clocks

This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.

Bits 11:8 **CR[3:0]**: CSR Clock Range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency (eth_hclk):

0000: MDC clock = eth_hclk / 42
0001: MDC clock = eth_hclk / 62
0010: MDC clock = eth_hclk / 16
0011: MDC clock = eth_hclk / 26
0100: MDC clock = eth_hclk / 102
0101: MDC clock = eth_hclk / 124
0110 to 0111: Reserved, must not be used

The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range.

When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits to 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3.

Program the following values only if the interfacing chips support faster MDC clocks:

1000: eth_hclk / 4
1001: eth_hclk / 6
1010: eth_hclk / 8
1011: eth_hclk / 10
1100: eth_hclk / 12
1101: eth_hclk / 14
1110: eth_hclk / 16
1111: eth_hclk / 18

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SKAP**: Skip Address Packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]**: GMII Operation Command

This bit indicates the operation command to the PHY.

00: Reserved, must not be used
01: Write
10: Post Read Increment Address for Clause 45 PHY
11: Read

When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E**: Clause 45 PHY Enable

When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **GB**: GMII Busy

The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in *MDIO address register (ETH_MACMDIOAR)* and *MDIO data register (ETH_MACMDIODR)* as long as this bit is set.

For write transfers, the application must first write 16-bit data in the GD field (and also RA field when C45E is set) in *MDIO data register (ETH_MACMDIODR)* register before setting this bit. When C45E is set, it should also write into the RA field of *MDIO data register (ETH_MACMDIODR)* before initiating a read transfer. When a read transfer is completed (GMII busy=0), the data read from the PHY register is valid in the GD field of the *MDIO data register (ETH_MACMDIODR)*.

Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit.

MDIO data register (ETH_MACMDIODR)

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in *MDIO address register (ETH_MACMDIOAR)* This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RA[15:0]**: Register Address

This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **GD[15:0]**: GMII Data

This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

ARP address register (ETH_MACARPAR)

Address offset: 0x0210

Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPPA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ARPPA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSWCR)

Address offset: 0x0230

Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bits clearing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SEEN**: Slave Error Response Enable

When this bit is set, the MAC responds with a Slave Error for accesses to reserved registers in CSR space.

When this bit is reset, the MAC responds with an Okay response to any register accessed from CSR space.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RCWE**: Register Clear on Write 1 Enable

When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.

When this bit is reset, the access mode to these register fields remains rc_r (clear on read).

MAC Address 0 high register (ETH_MACA0HR)

Address offset: 0x0300

Reset value: 0x8000 FFFF

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the GMII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x11 22 33 44 55 66 is received (0x11 in lane 0 of the first column) on the GMII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x66 55 44 33 22 11.

If the MAC address registers are configured to be double-synchronized to the GMII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable
This bit is always set to 1.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]
This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)

Reset value: 0xFFFF FFFF

The MAC Address x Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)
This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x high register (ETH_MACAxHR)

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC Address x High register holds the upper 16 bits of the second 6-byte MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable
When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30 **SA**: Source Address
When this bit is set, the MAC Addressx[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.
0: DA
1: SA

Bits 29:24 **MBC[5:0]**: Mask Byte Control
These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:
Bit 29: ETH_MACAxHR[15:8]
Bit 28: ETH_MACAxHR[7:0]
Bit 27: ETH_MACAxLR[31:24]
Bit 26: ETH_MACAxLR[23:16]
Bit 25: ETH_MACAxLR[15:8]
Bit 24: ETH_MACAxLR[7:0]
You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address1 [47:32]
This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the MMC operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of *Packet filtering control register (ETH_MACPFR)*.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF FFF0 (Full 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value.

Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Bit 2 **RSTONRD**: Reset on Read

When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover

When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)

Address offset: 0x0704

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Receive statistics counters.

The MMC receive interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, interrupts are set but the counter remains at all-ones. The MMC receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.
				rc_r	rc_r									rc_r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
									rc_r	rc_r					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC Receive LPI transition counter interrupt status

This bit is set when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC Receive LPI microsecond counter interrupt status

This bit is set when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIS**: MMC Receive Unicast Good Packet Counter Interrupt Status

This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC Receive Alignment Error Packet Counter Interrupt Status

This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC Receive CRC Error Packet Counter Interrupt Status

This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC transmit interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones.

The MMC transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.
				rc_r	rc_r					rc_r					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_r	rc_r														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIS**: MMC Transmit LPI transition counter interrupt status

This bit is set when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIS**: MMC Transmit LPI microsecond counter interrupt status

This bit is set when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIS**: MMC Transmit Good Packet Counter Interrupt Status

This bit is set when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIS**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Status

This bit is set when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIS**: MMC Transmit Single Collision Good Packet Counter Interrupt Status

This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C

Reset value: 0x0000 0000

The MMC receive interrupt mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.
				rw	rw									rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
									rw	rw					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC Receive LPI transition counter interrupt Mask

Setting this bit masks the interrupt when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC Receive LPI microsecond counter interrupt Mask

Setting this bit masks the interrupt when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIM**: MMC Receive Unicast Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIM**: MMC Receive Alignment Error Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC Receive CRC Error Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710

Reset value: 0x0000 0000

The MMC transmit interrupt mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32-bit wide.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.
				rw	rw					rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC Transmit LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC Transmit LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIM**: MMC Transmit Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIM**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC Transmit Single Collision Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXSNGLCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXSNGLCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets
This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMULTCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXMULTCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets
This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.

Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)

Address offset: 0x0768

Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good
This field indicates the number of good packets transmitted.

Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)

Address offset: 0x0794

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRCERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets
This field indicates the number of packets received with CRC error.

Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXALGNERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXALGNERR[31:0]**: Rx Alignment Error Packets
This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)

Address offset: 0x07C4

Reset value: 0x0000 0000

This register provides the number of good unicast packets received by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUCASTG[31:0]**: Rx Unicast Packets Good
This field indicates the number of good unicast packets received.

Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)

Address offset: 0x07EC

Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPIUSC[31:0]**: Tx LPI Microseconds Counter
This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPITRC[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the *LPI control and status register (ETH_MACLCSR)*), the counter is incremented.

Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4

Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPIUSC[31:0]**: Rx LPI Microseconds Counter

This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPITRC[31:0]**: Rx LPI Transition counter
This field indicates the number of times Rx LPI Entry has occurred.

L3 and L4 control 0 register (ETH_MACL3L4C0R)

Address offset: 0x0900

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0
										rw	rw	rw	rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM0**: Layer 4 Destination Port Inverse Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM0**: Layer 4 Destination Port Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM0**: Layer 4 Source Port Inverse Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM0**: Layer 4 Source Port Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN0**: Layer 4 Protocol Enable
When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.
The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer4 Address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 Address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A00[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A00[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A00[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 Address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A10[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A10[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A10[31:0]**: Layer 3 Address 1 Field
When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.
When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.
When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 Address 2 filter 0 register (ETH_MACL3A20R)

Address offset: 0x0918

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A20[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A20[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A20[31:0]**: Layer 3 Address 2 Field
When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.
When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.
When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

Layer3 Address 3 filter 0 register (ETH_MACL3A30R)

Address offset: 0x091C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A30[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A30[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A30[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

L3 and L4 control 1 register (ETH_MACL3L4C1R)

Address offset: 0x0930

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1
										rw	rw	rw	rw		rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM1**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM1 bit is set high.

Bit 20 **L4DPM1**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM1**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM1 bit is set high.

Bit 18 **L4SPM1**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN1**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:

0: No bits are masked

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

127: All bits except MSb are masked

This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match

Condition: IPv4 packets

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM1 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching. This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L4SP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A01[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A01[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field

When the L3PEN1 and L3SAM1bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A11[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A11[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 1 Register (ETH_MACL3A21R)

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A21[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L3A21[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A21[31:0]**: Layer 3 Address 2 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Layer3 address 3 filter 1 register (ETH_MACL3A31R)

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A31[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L3A31[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A31[31:0]**: Layer 3 Address 3 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Timestamp control Register (ETH_MACTSCR)

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]	
			rw				rw						rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	Res.	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **AV8021ASMEN**: AV 802.1AS Mode Enable

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When the PTP offload feature is enabled, for PTP offload, the transport-specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering

When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.

When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.

For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.

For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPSEL[1:0]**: Select PTP packets for Taking Snapshots

These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in *Table 495: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master

When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages

When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see *Table 495: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets

When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format

When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control

When this bit is set, the Timestamp Low register rolls over after 0x3B9A C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets

When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TSADDREG**: Update Addend Register

When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSUPDT**: Update Timestamp

When this bit is set, the system time is updated (added or subtracted) with the value specified in *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.

This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2 **TSINIT**: Initialize Timestamp

When this bit is set, the system time is initialized (overwritten) with the value specified in the *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.

This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update

When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp

When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.

On the Receive side, the MAC processes the 1588 packets only if this bit is set.

Subsecond increment register (ETH_MACSSIR)

Address offset: 0x0B04

Reset value: 0x0000 0000

In Coarse Update mode (bit TSCFUPDT in *Timestamp control Register (ETH_MACTSCR)*), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in *Timestamp control Register (ETH_MACTSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0 Reserved, must be kept at reset value.

System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08

Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TSS[31:0]**: Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C

Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSSS[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in *Timestamp control Register (ETH_MACTSCR)*, each bit represents 1 ns. The maximum value is 0x3B9A C9FF after which it rolls-over to zero.

System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10

Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in *Timestamp control Register (ETH_MACTSCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the seconds part of the update.

When ADDSUB is reset, this field must be programmed with the seconds part of the update value.

When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.

For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF FFFE (that is, $2^{32} – 2$).

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0 in *System time nanoseconds update register (ETH_MACSTNUR)*. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, $2^{32} – 1$) and the *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD SUB	TSSS[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADDSUB**: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field is the subseconds part of the update.

- ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.
- ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the *Timestamp control Register (ETH_MACTSCR)*.
- TSCTRLSSR field in the *Timestamp control Register (ETH_MACTSCR)*is 1:
  - The programmed value must be 10^9 – <subsecond value>.
  - Each bit represents 1 ns and the programmed value should not exceed 0x3B9A C9FF.
- TSCTRLSSR field in the *Timestamp control Register (ETH_MACTSCR)* is 0:
  - The programmed value must be 2^{31} - <subsecond_value>.
  - Each bit represents an accuracy of 0.46 ns.

For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF FFFF (that is, 2^{31} – 1), when TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset and 0x3B9A C9FF (that is, 10^9 –1), when TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, 2^{32} – 1).

Timestamp addend register (ETH_MACTSAR)

Address offset: 0x0B18

Reset value: 0x0000 0000

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the *Timestamp control Register (ETH_MACTSCR)*). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSAR[31:0]**: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Timestamp status register (ETH_MACTSSR)

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]			
		r	r	r	r	r	rc_r					rc_r	rc_r	rc_r	rc_r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
rc_r												rc_r	rc_r	rc_r	rc_r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **ATSNS[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

Bit 24 **ATSSTM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

Bit 16: Auxiliary trigger 0

Bit 17: Auxiliary trigger 1

Bit 18: Auxiliary trigger 2

Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15 **TXTSSIS**: Tx Timestamp Status Interrupt Status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the *Tx timestamp status seconds register (ETH_MACTXTSSSR)* is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 14:4 Reserved, must be kept at reset value.

Bit 3 **TSTRGTERR0**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR elapses (see *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **AUXTSTRIG**: Auxiliary Timestamp Trigger Snapshot

This bit is set high when the auxiliary snapshot is written to the FIFO.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **TSTARGT0**: Timestamp Target Time Reached

When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR registers (see *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 0 **TSSOVF**: Timestamp Seconds Overflow

When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 0xFFFF FFFF.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)

Address offset: 0x0B30

Reset value: 0x0000 0000

This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	TXTSSLO[30:16]														
r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXTSSLO[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed

When this bit is set, it indicates one of the following:

- The timestamp of the current packet is ignored if TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)* is reset
- The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)* is set.

Bits 30:0 **TXTSSLO[30:0]**: Transmit Timestamp Status Low

This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

Tx timestamp status seconds register (ETH_MACTXTSSSR)

Address offset: 0x0B34

Reset value: 0x0000 0000

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXTSSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High

This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

Auxiliary control register (ETH_MACACR)

Address offset: 0x0B40

Reset value: 0x0000 0000

The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATSEN3**: Auxiliary Snapshot 3 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 6 **ATSEN2**: Auxiliary Snapshot 2 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 5 **ATSEN1**: Auxiliary Snapshot 1 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear

When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

Auxiliary timestamp nanoseconds register (ETH_MACATSNR)

Address offset: 0x0B48

Reset value: 0x0000 0000

The *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*, along with *Auxiliary timestamp seconds register (ETH_MACATSSR)*, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in *Timestamp status register (ETH_MACTSSR)* indicate the fill-level of the FIFO. The top of the FIFO is removed only when *Auxiliary timestamp seconds register (ETH_MACATSSR)* is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXTSLO[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp

Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

Auxiliary timestamp seconds register (ETH_MACATSSR)

Address offset: 0x0B4C

Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUXTSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp
Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

Timestamp ingress asymmetric correction register (ETH_MACTSIACR)

Address offset: 0x0B50

Reset value: 0x0000 0000

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTIAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTIAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction
This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000.
The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Timestamp egress asymmetric correction register (ETH_MACTSEACR)

Address offset: 0x0B54

Reset value: 0x0000 0000

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTEAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTEAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction
This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}.
For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD 8000, which is the 2's complement of 0x0002 8000(2.5×2^{16}). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003 4CCC (3.3×2^{16}).

Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSIC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSIC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction
This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction
This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0 [1:0]		PPSEN 0	PPSCTRL[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Reserved, must not be used

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control

This field controls the frequency of the PPS output (eth_ptp_pps_out) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:

0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.

0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.

0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.

0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

..

1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (eth_ptp_pps_out) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- *When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.*
- *When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of*
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high).
- *When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of*
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.

PPS control register [alternate] (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Reserved, must not be used

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).

Bits 3:0 **PPSCMD[3:0]:** Flexible PPS Output (eth_ptp_pps_out) Control
Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all zero'. The following list describes the values of PPSCMD0:
0000: No Command
0001: START Single Pulse.
This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.
0010: START Pulse Train.
This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.
0011: Cancel START.
This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.
0100: STOP Pulse Train at time.
This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.
0101: STOP Pulse Train immediately.
This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).
0110: Cancel STOP Pulse train.
This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.
0111 to 1111: Reserved, must not be used

PPS target time seconds register (ETH_MACPPSTTSR)

Address offset: 0x0B80

Reset value: 0x0000 0000

The PPS output Target Time Seconds register, along with *PPS target time nanoseconds register (ETH_MACPPSTTNR)*, is used to schedule an interrupt event (Bit TSSOVF of *Timestamp status register (ETH_MACTSSR)*) when the system time exceeds the value programmed in these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register
This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the *PPS control register (ETH_MACPPSCR)*.

PPS target time nanoseconds register (ETH_MACPPSTTNR)

Address offset: 0x0B84

Reset value: 0x0000 0000

The PPS Target Time Nanoseconds register is present only when more than one Flexible PPS output is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRGTBUSY0	TTSL0[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy

The MAC sets this bit when the PPSCMD0 field in the *PPS control register (ETH_MACPPSCR)* is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

Bits 30:0 **TTSL0[30:0]**: Target Time Low for PPS Register

This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in *PPS control register (ETH_MACPPSCR)*.

When the TSCTRLSSR bit is set in the *Timestamp control Register (ETH_MACTSCR)*, this value should not exceed 0x3B9A C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of subsecond increment value.

PPS interval register (ETH_MACPPSIR)

Address offset: 0x0B88

Reset value: 0x0000 0000

The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

PPS width register (ETH_MACPPSWR)

Address offset: 0x0B8C

Reset value: 0x0000 0000

The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

Note: The value programmed in this register must be lesser than the value programmed in PPS interval register (ETH_MACPPSIR).

PTP Offload control register (ETH_MACPOCR)

Address offset: 0x0BC0

Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DN[7:0]								Res.	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]**: Domain Number
This field indicates the domain Number in which the PTP node is operating.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation
When this bit is set, the Delay Request and Delay response are not generated for received SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger
When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger
When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the ASYNCEN bit for this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer Transparent mode.

Bit 1 **ASYNCEN**: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Clock Master mode.

Bit 0 **PTOEN**: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

PTP Source Port Identity 0 Register (ETH_MACSPI0R)

Address offset: 0x0BC4

Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI0[31:0]**: Source Port Identity 0
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

PTP Source port identity 1 register (ETH_MACSPI1R)

Address offset: 0x0BC8

Reset value: 0x0000 0000

This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI1[31:0]**: Source Port Identity 1

This field indicates bits [63:32] of sourcePortIdentity of PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)

Address offset: 0x0BCC

Reset value: 0x0000 0000

This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SPI2[15:0]**: Source Port Identity 2

This field indicates bits [79:64] of sourcePortIdentity of PTP node.

Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0

Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval

This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:

Delay_Req to SYNC Ratio

In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.

0: DelayReq generated for every received SYNC

1: DelayReq generated every alternate reception of SYNC

2: for every 4 SYNC messages

3: for every 8 SYNC messages

4: for every 16 SYNC messages

5: for every 32 SYNC messages

Others: Reserved, must not be used

The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:

Log Sync Interval

This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Ethernet MAC register map and reset values

Table 543. Ethernet MAC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	ETH_MACCR	ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE	PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0004	ETH_MACECR	Res	Res	EIPG[4:0]					EIPGEN	Res	Res	Res	Res	Res	USP	SPEN	DCRCC	Res	Res	GPSL[13:0]													
	Reset value			0	0	0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	ETH_MACPFR	RA	Res	Res	Res	Res	Res	Res	Res	Res	Res	DNTU	IPFE	Res	Res	Res	VTFE	Res	Res	Res	Res	Res	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
	Reset value	0										0	0				0						0	0	0	0	0	0	0	0	0	0	0
0x000C	ETH_MACWTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PWE	Res	Res	Res	Res	WTO[3:0]			
	Reset value																								0					0	0	0	0
0x0010	ETH_MACHT0R	HT31T0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	ETH_MACHT1R	HT63T32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018 - 0x004C	Reserved																																
0x0050	ETH_MACVTR	EIVLRXS	Res	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV	VL[15:0]															
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	Reserved																																
0x0058	ETH_MACVHTR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	VLHT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	Reserved																																
0x0060	ETH_MACVIR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0064	ETH_MACiVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068 - 0x006C	Reserved																																
0x0070	ETH_ MACQ0TXFCR	PT[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0			0	0
0x0074 - 0x008C	Reserved																																
0x0090	ETH_MACRXFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
	Reset value																															0	0
0x0094	ETH_MACRXQCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VFFQ	VFFQE	Res.	Res.	Res.	Res.	Res.	Res.	MFFQ	MFFQE	Res.	Res.	Res.	Res.	Res.	Res.	UFFQ	UFFQE
	Reset value															0	0							0	0							0	0
0x0098- 0x009C	Reserved																																
0x00A0	ETH_ MACRXQC0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ1EN[1:0]		RXQ0EN[1:0]	
	Reset value																													0	0	0	0
0x00A4	ETH_ MACRXQC1R	Res.	Res.	Res.	Res.	Res.	Res.			TPQC[1]	TPQC[0]	TACPQE	MCBCQEN	Res.	MCBCQ[2:0]			Res.	UPQ[2:0]			Res.	Res.	Res.	Res.	Res.	PTPQ[2:0]			Res.	AVCPQ[2:0]		
	Reset value									0	0	0	0		0	0	0		0	0	0						0	0	0		0	0	0
0x00A8	ETH_ MACRXQC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSRQ1[7:0]								PSRQ0[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	Reserved																																
0x00B0	ETH_MACISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	RGSMIIIS
	Reset value																		0	0	0		0	0	0			0	0	0			0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00B4	ETH_MACIER	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RXSTSIE	TXSTSIE	TSIE	Res	Res	Res	Res	Res	Res	LPIIE	PMTIE	PHYIE	Res	Res	RGSMIIIE
	Reset value																		0	0	0							0	0	0			0
0x00B8	ETH_ MACRXTXSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RWT	Res	Res	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
	Reset value																								0			0	0	0	0	0	0
0x00BC	Reserved																																
0x00C0	ETH_MACPCSR	RWKFILTRST	Res	Res	RWKPTR[4:0]					Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RWKPFE	GLBLUCAST	Res	Res	RWKPRCVD	MGKPRCVD	Res	Res	RWKPKTEN	MGKPKTEN	PWRDWN
	Reset value	0			0	0	0	0	0														0	0			0	0			0	0	0
0x00C4	ETH_ MACRWKPFR	MACRWKPFR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8 - 0x00CC	Reserved																																
0x00D0	ETH_MACLCSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LPITCSE	LPITE	LPITXA	PLSEN	PLS	LPIEN	Res	Res	Res	Res	Res	Res	RLPIST	TLPIST	Res	Res	Res	Res	RLPIEX	RLPIEN	TLPIEX	TLPIEN
	Reset value											0	0	0	0	0	0							0	0					0	0	0	0
0x00D4	ETH_MACLTCR	Res	Res	Res	Res	Res	Res	LST[9:0]										TWT[15:0]															
	Reset value							1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	ETH_MACLETR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LPIET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	ETH_ MAC1USTCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TIC_1US_CNTR[11:0]											
	Reset value																					0	0	0	0	0	1	1	0	0	0	1	1
0x00E0 - 0x00F4	Reserved																																
0x00F8	ETH_ MACPHYCSR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LNKSTS	LNKSPEED[1:0]		LNKMOD	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LUD	TC
	Reset value													0	0	0	0															0	0
0x00FC - 0x010C	Reserved																																

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0110	ETH_MACVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERVER[7:0]								SNPSVER[7:0]							
	Reset value	0x0000 4042																															
0x0114	ETH_MACDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
	Reset value														0	0	0														0	0	0
0x0118	Reserved																																
0x011C	ETH_MACHWF0R		ACTPHYSEL[3:0]			SAVLANINS	TSSTSSEL[1:0]		MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL	Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	Reset value	0x0A0D 73F7																															
0x0120	ETH_MACHWF1R	Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN	ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					Res.	RXFIFOSIZE[4:0]				
	Reset value	0x1114 1945																															
0x0124	ETH_MACHWF2R	Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]		RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
	Reset value	0x4104 0041																															
0x0128	ETH_MACHWF3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	Reset value	0x0000 0020																															
0x012C - 0x01FC	Reserved																																
0x0200	ETH_ MACMDIOAR	Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]					Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	GB
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0204	ETH_ MACMDIODR	RA[15:0]																GD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0208 - 0x020C	Reserved																																
0x0210	ETH_MACARPAR	ARPPA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0214 - 0x022C	Reserved																																
0x0230	ETH_ MACCSRSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
	Reset value																								0								0
0x0234- 0x23C	Reserved																																
0x0240 - 0x02FC	Reserved																																
0x0300	ETH_MACA0HR	AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	1																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0304	ETH_MACA0LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0308	ETH_MACA1HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30C	ETH_MACA1LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0310	ETH_MACA2HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0314	ETH_MACA2LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0318	ETH_MACA3HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x031C	ETH_MACA3LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0320 - 0x06FC	Reserved																																
0x0700	ETH_MMC_ CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
	Reset value																								0			0	0	0	0	0	0
0x0704	ETH_MMC_RX_ INTERRUPT	Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0708	ETH_MMC_TX_ INTERRUPT	Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x070C	ETH_MMC_RX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0710	ETH_MMC_TX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x0714 - 0x0748	Reserved																																

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x074C	ETH_TX_SINGLE_COLLISION_GOOD_PACKETS	TXSNGLCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0750	ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS	TXMULTCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0754 - 0x0764	Reserved																																
0x0768	ETH_TX_PACKET_COUNT_GOOD	TXPKTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x076C - 0x0790	Reserved																																
0x0794	ETH_RX_CRC_ERROR_PACKETS	RXCRCERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0798	ETH_RX_ALIGNMENT_ERROR_PACKETS	RXALGNERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x079C - 0x07C0	Reserved																																
0x07C4	ETH_RX_UNICAST_PACKETS_GOOD	RXUCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C8 - 0x07E8	Reserved																																
0x07EC	ETH_TX_LPI_USEC_CNTR	TXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F0	ETH_TX_LPI_TRAN_CNTR	TXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F4	ETH_RX_LPI_USEC_CNTR	RXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F8	ETH_RX_LPI_TRAN_CNTR	RXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x07FC-0x8FC	Reserved																																
0x0900	ETH_ MACL3L4C0R	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res	L4PEN0	L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res	L3PEN0
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0904	ETH_MACL4A0R	L4DP0[15:0]																L4SP0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0908 - 0x090C	Reserved																																
0x0910	ETH_MACL3A00R	L3A00[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0914	ETH_MACL3A10R	L3A10[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0918	ETH_MACL3A20R	L3A20[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x091C	ETH_MACL3A30R	L3A30[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0920 - 0x092C	Reserved																																
0x0930	ETH_ MACL3L4C1R	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res	L4PEN1	L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res	L3PEN1
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0934	ETH_MACL4A1R	L4DP1[15:0]																L4SP1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0938 - 0x093C	Reserved																																
0x0940	ETH_MACL3A01R	L3A01[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0944	ETH_MACL3A11R	L3A11[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0948	ETH_MACL3A21R	L3A21[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094C	ETH_MACL3A31R	L3A31[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0950 - 0x0AFC	Reserved																																
0x0B00	ETH_MACTSCR	Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]		TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	Res.	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
	Reset value				0				0						0	0	0	0	0	1	0	0	0	0	0			0		0	0	0	0
0x0B04	ETH_MACSSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x0B08	ETH_MACSTSR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0C	ETH_MACSTNR	Res.	TSSS[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B10	ETH_MACSTSUR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B14	ETH_MACSTNUR	ADDSUB	TSSS[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B18	ETH_MACTSAR	TSAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B1C	Reserved																																
0x0B20	ETH_MACTSSR	Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]				TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
	Reset value			0	0	0	0	0	0					0	0	0	0	0												0	0	0	0
0x0B24-0xB28	Reserved																																
0x0B2C	Reserved																																
0x0B30	ETH_MACTXTSSNR	TXTSSMIS	TXTSSLO[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B34	ETH_ MACTXTSSSR	TXTSSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B38 - 0x0B3C	Reserved																																
0x0B40	ETH_MACACR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res	Res	Res	ATSFC
	Reset value																									0	0	0	0				0
0x0B44	Reserved																																
0x0B48	ETH_MACATSNR	Res	AUXTSLO[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4C	ETH_MACATSSR	AUXTSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B50	ETH_MACTSIACR	OSTIAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B54	ETH_MACTSEACR	OSTEAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B58	ETH_ MACTSICNR	TSIC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B5C	ETH_ MACTSECNR	TSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B60 - 0x0B6C	Reserved																																
0x0B70	ETH_MACPPSCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
	Reset value																										0	0	0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B70	ETH_MACPPSCR (alternate)	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
	Reset value																										0	0	0	0	0	0	0
0x0B74 - 0x0B7C	Reserved																																
0x0B80	ETH_ MACPPSTTSR	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B84	ETH_ MACPPSTTNR	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B88	ETH_MACPPSIR	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8C	ETH_MACPPSWR	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B90 - 0x0BBC	Reserved																																
0x0BC0	ETH_MACPOCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DN[7:0]								Res	DRRDIS	APDREQTRIG	ASYNCTRIG	Res	APDREQEN	ASYNCEN	PTOEN
	Reset value																	0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x0BC4	ETH_MACSPI0R	SPI0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC8	ETH_MACSPI1R	SPI1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BCC	ETH_MACSPI2R	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SPI2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 543. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0BD0	ETH_MACLMIR	LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
	Reset value	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5 on page 153* for the register boundary addresses.

62 HDMI-CEC controller (CEC)

62.1 HDMI-CEC introduction

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

62.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 specification
- Independent 32 kHz CEC kernel (refer to *section RCC kernel clock distribution*)
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - with automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)

62.3 HDMI-CEC functional description

62.3.1 HDMI-CEC pin and internal signals

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on *Figure 772*.

Table 545 lists the internal signals that are exchanged between the HDMI-CEC and other functional blocks (such as RCC and EXTI).

Table 544. HDMI pin

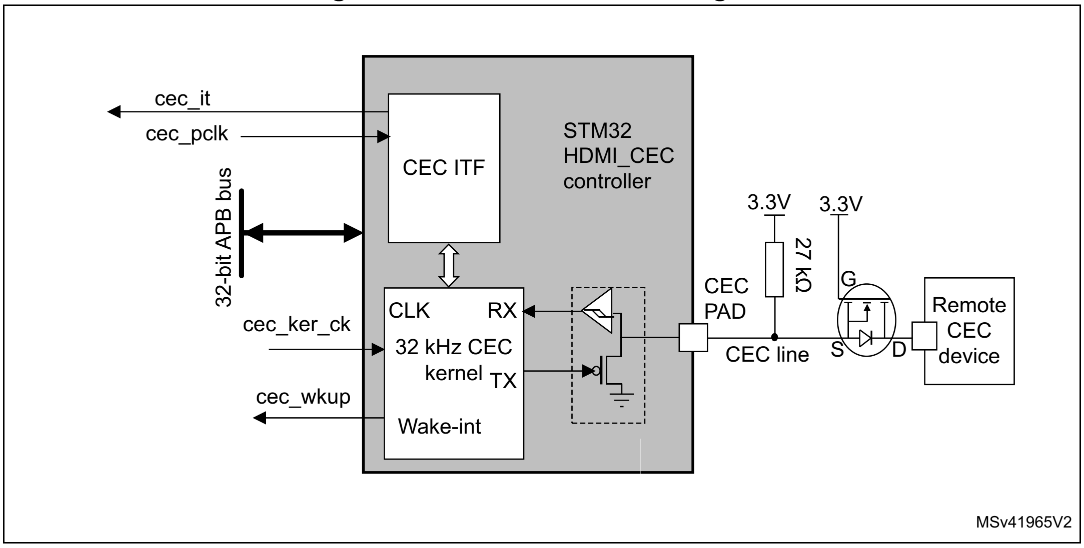
Name	Signal type	Remarks
CEC	Bidirectional	Two states: – 1 = high impedance – 0 = low impedance A 27 kΩ resistor must be added externally.

Table 545. HDMI-CEC internal input/output signals

Signal name	Signal type	Description
cec_wkup	Digital output	HDMI-CEC wake-up signal
cec_it	Digital output	HDMI-CEC interrupt signal
cec_pclk	Digital input	APB clock
cec_ker_ck	Digital input	HDMI-CEC kernel clock

62.3.2 HDMI-CEC block diagram

Figure 772. HDMI-CEC block diagram



62.3.3 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to ‘ping’ other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

Figure 773. Message structure

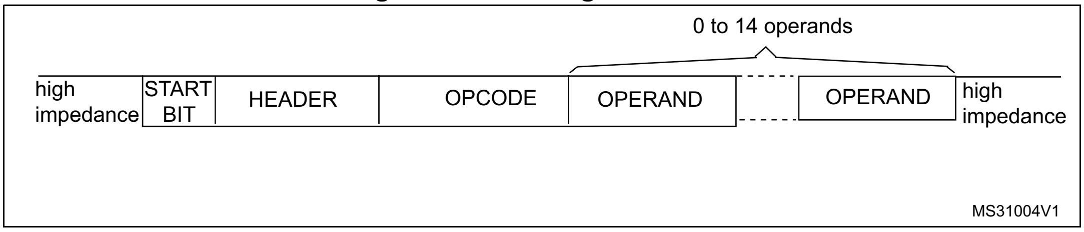
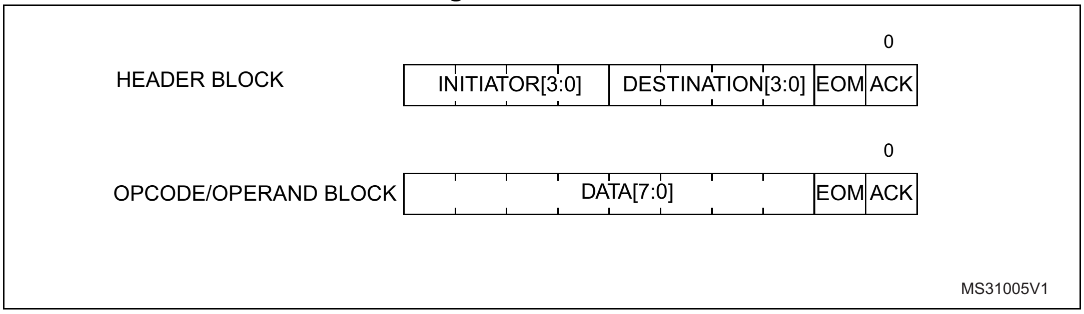


Figure 774. Blocks

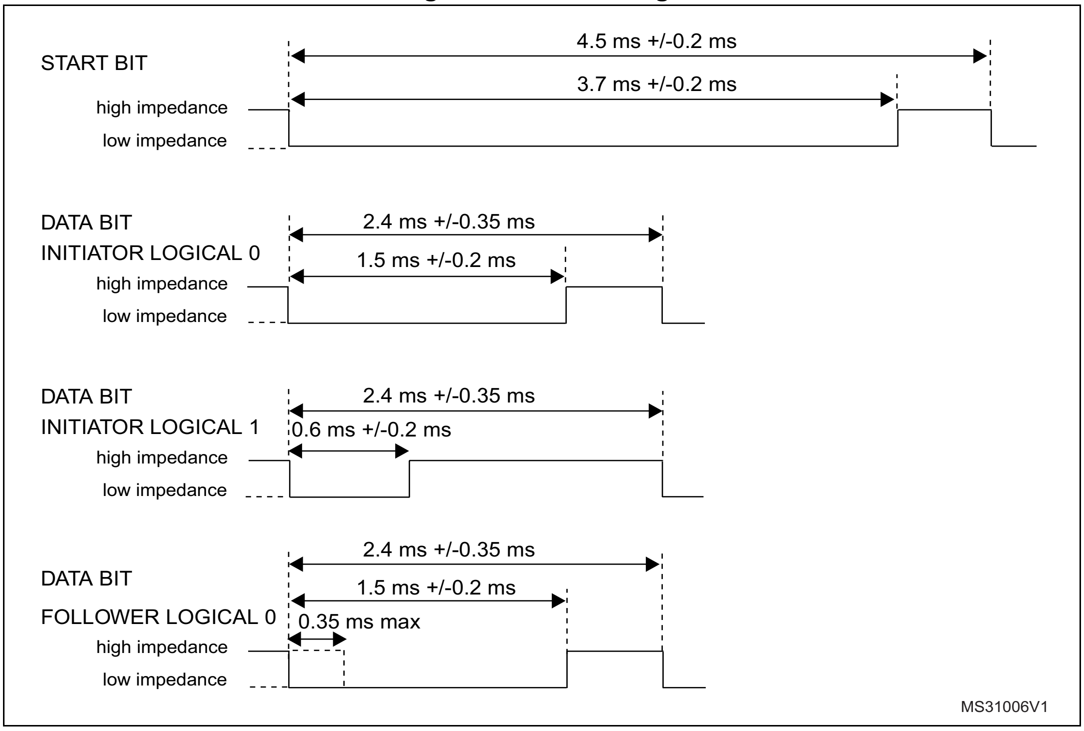


62.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

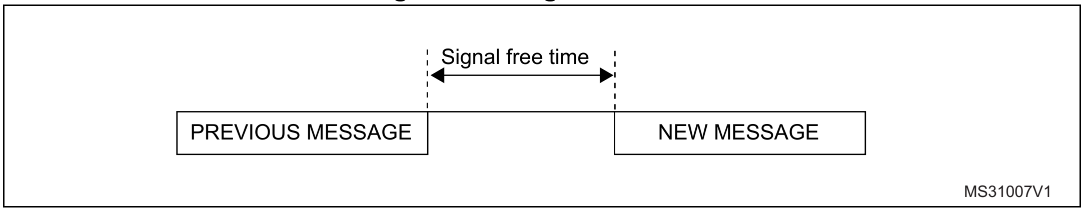
Figure 775. Bit timings



62.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the figure below.

Figure 776. Signal free time



Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator must monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0. Assuming then it has lost arbitration, it stops transmitting and becomes a follower.

Figure 777. Arbitration phase

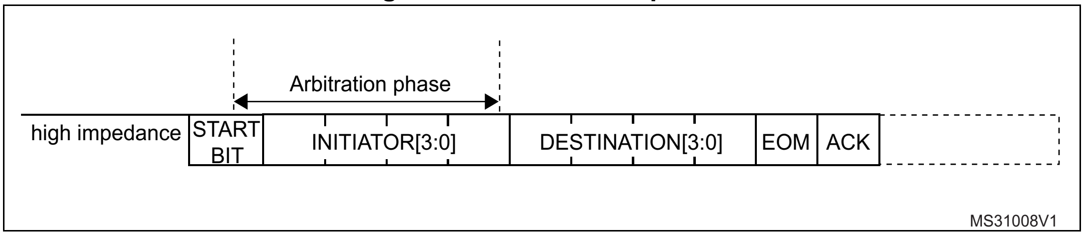
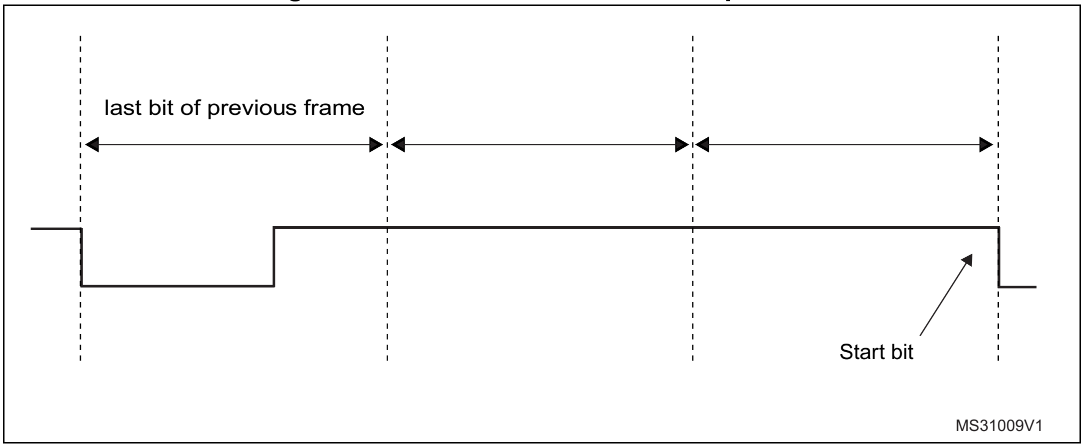


Figure 778 shows an example for a SFT of three nominal bit periods.

Figure 778. SFT of three nominal bit periods



A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission

This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

62.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM = 1).

In case of SFTOPT = 1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

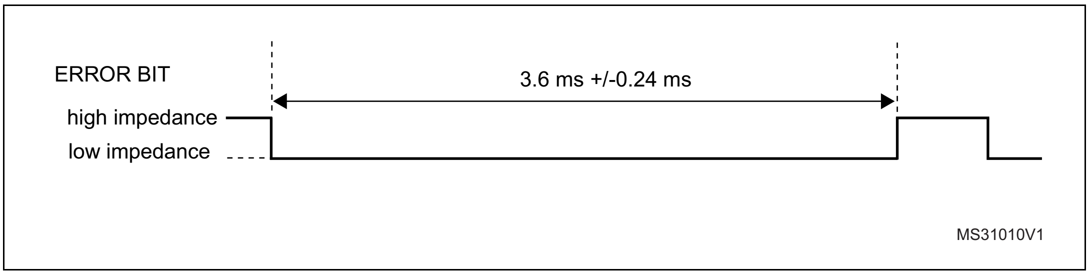
- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR = 1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACKE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncodified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

62.5 Error handling

62.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period (3.6 ms nominally).

Figure 779. Error bit timing



62.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

62.5.3 Bit rising error (BRE)

BRE (bit rising error) is set when a bit rising edge is detected outside the windows where it is expected (see *Figure 780*). BRE flag also generates a CEC interrupt if the BREIE = 1.

In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP = 1 an error bit is generated even if BREGEN = 0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN = 0, BRDNOGEN = 1.

62.5.4 Short bit period error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see *Figure 780*). SBPE flag also generates a CEC interrupt if the SBPEIE = 1.

An error bit is always generated on the line in case of a SBPE error detection. An error bit is not generated upon SBPE detection only when Listen mode is set (LSTN = 1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN = 1

62.5.5 Long bit period error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see *Figure 780*). LBPE flag also generates a CEC interrupt if the LBPEIE = 1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN = 0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN = 0, BRDNOGEN = 1.

Note: The BREGEN = 1, BRESTP = 0 configuration must be avoided.

Figure 780. Error handling

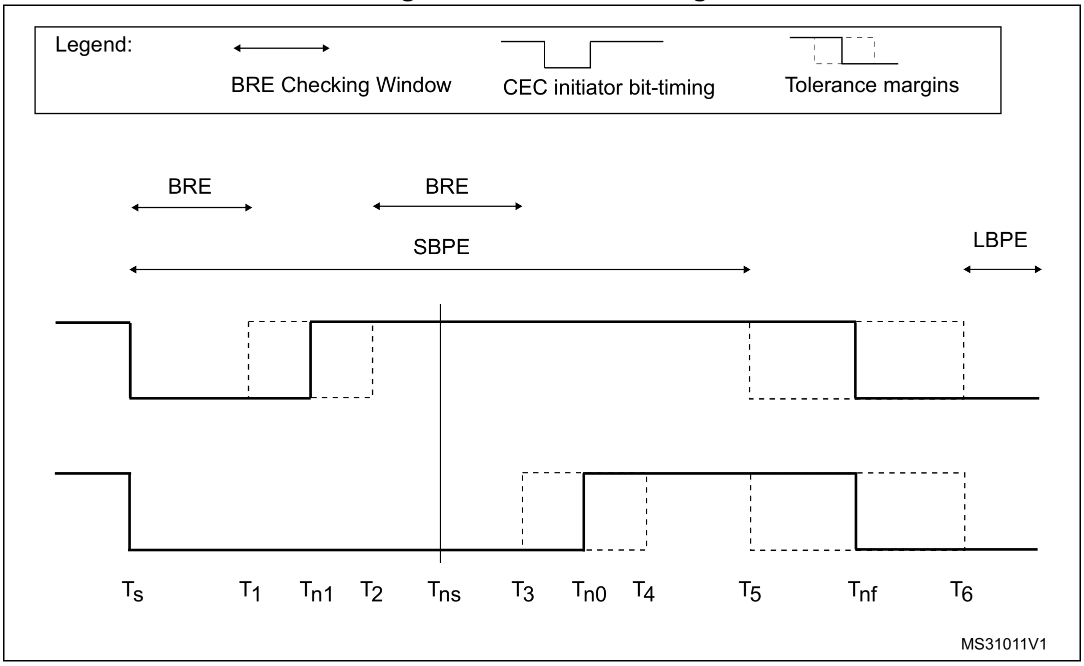


Table 546. Error handling timing parameters

Time	RXTOL	ms	Description
T_s	x	0	Bit start event.
T_1	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T_{n1}	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T_2	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T_{ns}	x	1.05	Nominal sampling time.
T_3	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	
T_{n0}	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T_4	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T_5	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T_{nf}	x	2.4	The nominal data bit period.
T_6	0	2.75	The latest time for the start of a following bit.
	1	2.95	

62.5.6 Transmission error detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE = 1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to five times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

Figure 781. TXERR detection

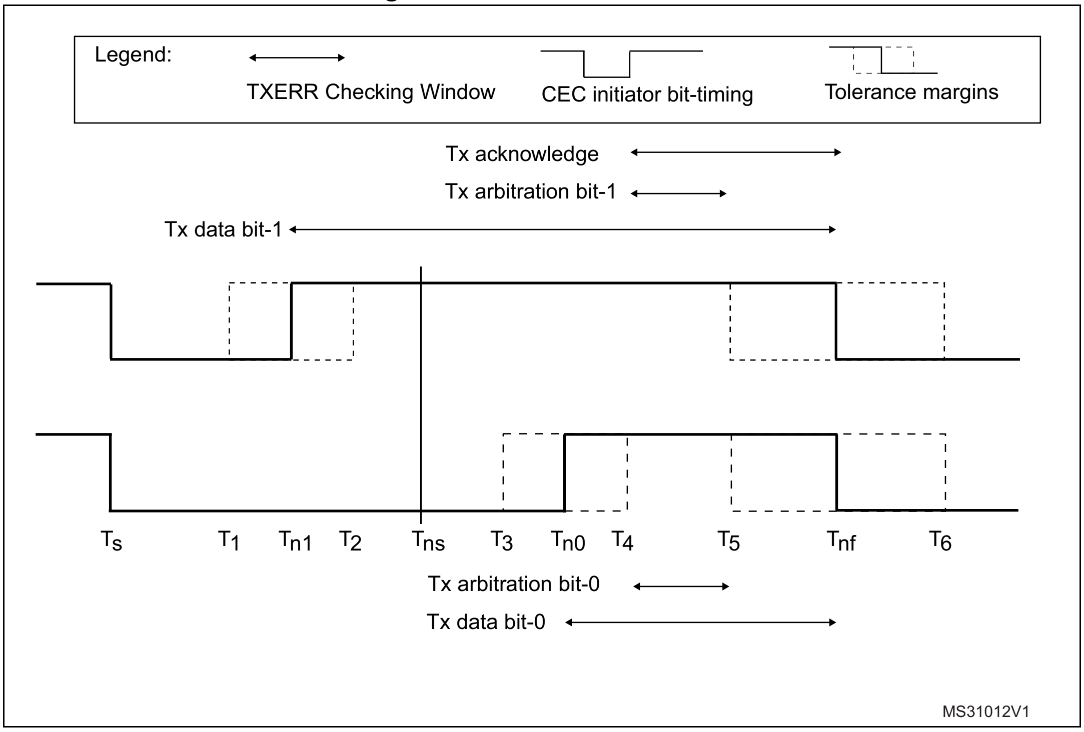


Table 547. TXERR timing parameters

Time	RXTOL	ms	Description
T_s	x	0	Bit start event.
T_1	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T_{n1}	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T_2	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T_{ns}	x	1.05	Nominal sampling time.
T_3	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	

Table 547. TXERR timing parameters (continued)

Time	RXTOL	ms	Description
T_{n0}	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T_4	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T_5	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T_{nf}	x	2.4	The nominal data bit period.
T_6	0	2.75	The latest time for the start of a following bit.
	1	2.95	

62.6 HDMI-CEC interrupts

An interrupt can be produced:

- during reception if a receive block transfer is finished or if a receive error occurs.
- during transmission if a transmit block transfer is finished or if a transmit error occurs.

Table 548. HDMI-CEC interrupts

Interrupt event	Event flag	Enable control bit
Rx-byte received	RXBR	RXBRIE
End of reception	RXEND	RXENDIE
Rx-overrun	RXOVR	RXOVRIE
Rxbit rising error	BRE	BREIE
Rx-short bit period error	SBPE	SBPEIE
Rx-long bit period error	LBPE	LBPEIE
Rx-missing acknowledge error	RXACKE	RXACKEIE
Arbitration lost	ARBLST	ARBLSTIE
Tx-byte request	TXBR	TXBRIE
End of transmission	TXEND	TXENDIE
Tx-buffer underrun	TXUDR	TXUDRIE
Tx-error	TXERR	TXERRIE
Tx-missing acknowledge error	TXACKE	TXACKEIE

62.7 HDMI-CEC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions. The registers have to be accessed by words (32 bits).

62.7.1 CEC control register (CEC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEOM	TXSOM	CECEN
													rs	rs	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TXEOM**: Tx end of message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message. TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.

0: TXDR data byte is transmitted with EOM = 0.

1: TXDR data byte is transmitted with EOM = 1.

Note: TXEOM must be set when CECEN = 1.

TXEOM must be set before writing transmission data to TXDR.

If TXEOM is set when TXSOM = 0, transmitted message consists of 1 byte (HEADER) only (PING message).

Bit 1 **TXSOM**: Tx start of message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.

Start-bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission starts after the end of reception.

TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND = 1), in case of transmission underrun (TXUDR = 1), negative acknowledge (TXACKE = 1), and transmission error (TXERR = 1). It is also cleared by CECEN = 0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST = 1).

TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.

0: No CEC transmission is on-going.

1: CEC transmission command

Note: TXSOM must be set when CECEN = 1.

TXSOM must be set when transmission data is available into TXDR.

HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC_CFGR.OAR that is used only for reception.

Bit 0 **CECEN:** CEC enable

The CECEN bit is set and cleared by software. CECEN = 1 starts message reception and enables the TXSOM control. CECEN = 0 disables the CEC peripheral, clears all bits of CEC_CR register and aborts any on-going reception or transmission.

0: CEC peripheral is off.

1: CEC peripheral is on.

62.7.2 CEC configuration register (CEC_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x04

Reset value: 0x0000 0000

Caution: It is mandatory to write CEC_CFGR only when CECEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSTN	OAR[14:0]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOP	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LSTN:** Listen mode

LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

Bits 30:16 **OAR[14:0]:** Own addresses configuration

The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.

At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN = 1), but without acknowledge sent. Broadcast messages are always received.

Example:

OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5. Consequently, each message directed to one of these addresses is received.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **SFTOP**: SFT option bit

The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software.

1: SFT timer starts automatically at the end of message transmission/reception.

Bit 7 **BRDNOGEN:** Avoid error-bit generation in broadcast

The BRDNOGEN bit is set and cleared by software.

0: BRE detection with BRESTP = 1 and BREGEN = 0 on a broadcast message generates an error-bit on the CEC line. LBPE detection with LBPEGEN = 0 on a broadcast message generates an error-bit on the CEC line.

1: Error-bit is not generated in the same condition as above. An error-bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6 **LBPEGEN:** Generate error-bit on long bit period error

The LBPEGEN bit is set and cleared by software.

0: LBPE detection does not generate an error-bit on the CEC line.

1: LBPE detection generates an error-bit on the CEC line.

Note: If BRDNOGEN = 0, an error-bit is generated upon LBPE detection in broadcast even if LBPEGEN = 0.

Bit 5 **BREGEN:** Generate error-bit on bit rising error

The BREGEN bit is set and cleared by software.

0: BRE detection does not generate an error-bit on the CEC line.

1: BRE detection generates an error-bit on the CEC line (if BRESTP is set).

Note: If BRDNOGEN = 0, an error-bit is generated upon BRE detection with BRESTP = 1 in broadcast even if BREGEN = 0.

Bit 4 **BRESTP**: Rx-stop on bit rising error

The BRESTP bit is set and cleared by software.

0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.

1: BRE detection stops message reception.

Bit 3 **RXTOL:** Rx-tolerance

The RXTOL bit is set and cleared by software.

0: Standard tolerance margin:

– Start-bit, +/- 200 µs rise, +/- 200 µs fall
– Data-bit: +/- 200 µs rise. +/- 350 µs fall

1: Extended tolerance

– Start-bit: +/- 400 µs rise, +/- 400 µs fall
– Data-bit: +/-300 µs rise, +/- 500 µs fall

Bits 2:0 **SFT[2:0]**: Signal free time

SFT bits are set by software. In the SFT = 0x0 configuration, the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.

0x0

– 2.5 data-bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST = 1, TXERR = 1, TXUDR = 1 or TXACKE = 1)
– 4 data-bit periods if CEC is the new bus initiator
– 6 data-bit periods if CEC is the last bus initiator with successful transmission (TXEOM = 1)

0x1: 0.5 nominal data bit periods
0x2: 1.5 nominal data bit periods
0x3: 2.5 nominal data bit periods
0x4: 3.5 nominal data bit periods
0x5: 4.5 nominal data bit periods
0x6: 5.5 nominal data bit periods
0x7: 6.5 nominal data bit periods

62.7.3 CEC Tx data register (CEC_TXDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXD[7:0]**: Tx data

TXD is a write-only register containing the data byte to be transmitted.

62.7.4 CEC Rx data register (CEC_RXDR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXD[7:0]**: Rx data

RXD is read-only and contains the last data byte that has been received from the CEC line.

62.7.5 CEC interrupt and status register (CEC_ISR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACKE	TXERR	TXUDR	TXEND	TXBR	ARBLST	RXACKE	LBPE	SBPE	BRE	RXOVR	RXEND	RXBR
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKE:** Tx-missing acknowledge error

In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.

TXACKE is cleared by software write at 1.

Bit 11 **TXERR:** Tx-error

In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.

TXERR is cleared by software write at 1.

Bit 10 **TXUDR:** Tx-buffer underrun

In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.

TXUDR is cleared by software write at 1

Bit 9 **TXEND:** End of transmission

TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.

TXEND is cleared by software write at 1.

Bit 8 **TXBR:** Tx-byte request

TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within six nominal data-bit periods before transmission underrun error occurs (TXUDR).

TXBR is cleared by software write at 1.

Bit 7 **ARBLST:** Arbitration lost

ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.

ARBLST is cleared by software write at 1.

Bit 6 **RXACKE:** Rx-missing acknowledge

In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.

RXACKE is cleared by software write at 1.

Bit 5 **LBPE:** Rx-long bit period error

LBPE is set by hardware in case a data-bit waveform is detected with long bit period error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an error-bit on the CEC line if LBPEGEN = 1. In case of broadcast, error-bit is generated even in case of LBPEGEN = 0.

LBPE is cleared by software write at 1.

Bit 4 **SBPE:** Rx-short bit period error

SBPE is set by hardware in case a data-bit waveform is detected with short bit period error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an error-bit on the CEC line.

SBPE is cleared by software write at 1.

Bit 3 **BRE**: Rx-bit rising error

BRE is set by hardware in case a data-bit waveform is detected with bit rising error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL, in case rising edge is still longing. BRE stops message reception if BRESTP = 1. BRE generates an error-bit on the CEC line if BREGEN = 1.

BRE is cleared by software write at 1.

Bit 2 **RXOVR**: Rx-overrun

RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent.

RXOVR is cleared by software write at 1.

Bit 1 **RXEND**: End of reception

RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR.

RXEND is cleared by software write at 1.

Bit 0 **RXBR**: Rx-byte received

The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer.

RXBR is cleared by software write at 1.

62.7.6 CEC interrupt enable register (CEC_IER)

Address offset: 0x14

Reset value: 0x0000 0000

Caution: It is mandatory to write CEC_IER only when CECEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKIE**: Tx-missing acknowledge error interrupt enable

The TXACKEIE bit is set and cleared by software.

0: TXACKE interrupt disabled

1: TXACKE interrupt enabled

Bit 11 **TXERRIE**: Tx-error interrupt enable

The TXERRIE bit is set and cleared by software.

0: TXERR interrupt disabled

1: TXERR interrupt enabled

Bit 10 **TXUDRIE**: Tx-underrun interrupt enable

The TXUDRIE bit is set and cleared by software.

0: TXUDR interrupt disabled

1: TXUDR interrupt enabled

Bit 9 **TXENDIE**: Tx-end of message interrupt enable
The TXENDIE bit is set and cleared by software.
0: TXEND interrupt disabled
1: TXEND interrupt enabled

Bit 8 **TXBRIE**: Tx-byte request interrupt enable
The TXBRIE bit is set and cleared by software.
0: TXBR interrupt disabled
1: TXBR interrupt enabled

Bit 7 **ARBLSTIE**: Arbitration lost interrupt enable
The ARBLSTIE bit is set and cleared by software.
0: ARBLST interrupt disabled
1: ARBLST interrupt enabled

Bit 6 **RXACKIE**: Rx-missing acknowledge error interrupt enable
The RXACKIE bit is set and cleared by software.
0: RXACKE interrupt disabled
1: RXACKE interrupt enabled

Bit 5 **LBPEIE**: Long bit period error interrupt enable
The LBPEIE bit is set and cleared by software.
0: LBPE interrupt disabled
1: LBPE interrupt enabled

Bit 4 **SBPEIE**: Short bit period error interrupt enable
The SBPEIE bit is set and cleared by software.
0: SBPE interrupt disabled
1: SBPE interrupt enabled

Bit 3 **BREIE**: Bit rising error interrupt enable
The BREIE bit is set and cleared by software.
0: BRE interrupt disabled
1: BRE interrupt enabled

Bit 2 **RXOVRIE**: Rx-buffer overrun interrupt enable
The RXOVRIE bit is set and cleared by software.
0: RXOVR interrupt disabled
1: RXOVR interrupt enabled

Bit 1 **RXENDIE**: End of reception interrupt enable
The RXENDIE bit is set and cleared by software.
0: RXEND interrupt disabled
1: RXEND interrupt enabled

Bit 0 **RXBRIE**: Rx-byte received interrupt enable
The RXBRIE bit is set and cleared by software.
0: RXBR interrupt disabled
1: RXBR interrupt enabled

62.7.7 HDMI-CEC register map

Table 549. HDMI-CEC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEOM	TXSOM	CECEN
	Reset value																														0	0	0
0x04	CEC_CFGR	LSTN	OAR[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOPT	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x08	CEC_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C	CEC_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x10	CEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKE	TXERR	TXUDR	TXEND	TXBR	ARBLST	RXACKE	LBPE	SBPE	BRE	RXOVR	RXEND	RXBR
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	CEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.5* for the register boundary addresses.

63 Hardware debug port (HDP)

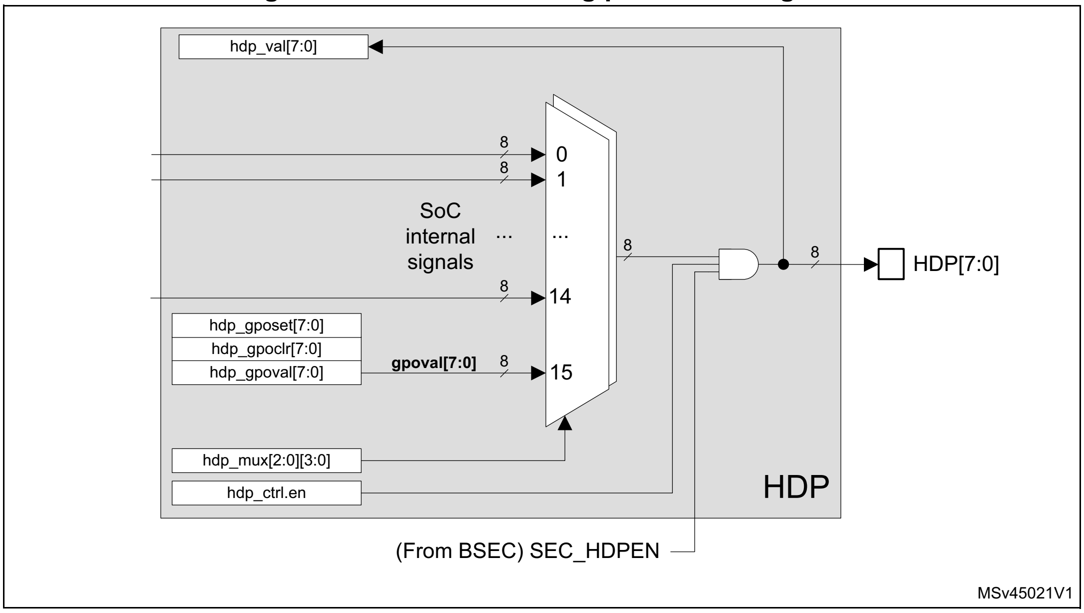
The Hardware Debug Port allows the observation of internal signals. By using multiplexers, up to 16 signals for each of 8-bit output can be observed.

63.1 Features

- 8 output signals
- One of 15 internal signals with individual control
- 8 Software programmable signals for pinout agnostic code debugging
- Output disabling by security signal

63.2 Block diagram

Figure 782. Hardware debug port block diagram



63.3 Functional description

The main multiplexer, allowing the selection of the 8-signal debug set to be output, is controlled through the HDP_CTRL register.

The HDP value at the output of the main multiplexer can be read by software through the HDP_VAL register.

During debug it is often useful to output some signals that are controlled by software; these can be used to observe a precise code sequence or to indicate the current executed task. Debug set #15 is reserved for this purpose; it provides on HDP outputs the value of the HDP_GPOVAL register. This register is associated with set and clear registers (HDP_GPOSET and HDP_GPOCLR) allowing the signals to be toggled without a read-modify-write sequence.

Some debug sets expose internal data that require protection; the SEC_HDPEN signal, when low, forces all the HDP block outputs low.

Note: The internal signals may toggle at a higher rate than the pads can support, in which case their observability may be compromised. The user should be aware of this limitation when using the HDP port.

Table 550. HDP signal multiplexing

HDP_MUX	HDP7	HDP6	HDP5	HDP4	HDP3	HDP2	HDP1	HDP0
	MUX7[3:0]	MUX6[3:0]	MUX5[3:0]	MUX4[3:0]	MUX3[3:0]	MUX2[3:0]	MUX1[3:0]	MUX0[3:0]
0	CA7 STANDBYWFI0	CA7 STANDBYWFI1	CA7 STANDBYWFIL2	PWR (=PWR_MPUCR. PDDS and not(PWR_MPUC R.CSTBYDIS))	PWR SEL_VTH_VDD CORE	PWR pwrwake_mpu	PWR pwrwake_mcu	PWR pwrwake_sys
1	CA7 STANDBYWFE0	CA7 STANDBYWFE1	PWR VTH_VDDCORE _ACK	CM4 SLEEPING	CM4 TXEV	CM4 RXEV	CM4 HALTED	CM4 SLEEPDEEP
2	-	CA7 EVENTO	CA7 nRESET0	CA7 nRESET1	CA7 nPMUIRQ0	CA7 nPMUIRQ1	CA7 nAXIERRIRQ	PWR STDBY_WKUP
3	CA7 DBGACK0	CA7 DBGACK1	CA7 nIRQOUT0	CA7 nIRQOUT1	CA7 nFIQOUT0	CA7 nFIQOUT1	PWR OKIN_MR	PWR ENCOMP_ VDDCORE
4	BSEC out fuse_ok	-	BSEC in pwrok	BSEC out sec_dften	BSEC out sec_dftlock	BSEC in rstcore_n	BSEC out sec_dbgen	BSEC out sec_niden
5	BSEC out sec_spiden	BSEC out sec_spniden	BSEC out sec_deviceen	BSEC out sec_dbgswenable	EXTI c1_wakeup	EXTI c2_wakeup	EXTI sys_wakeup	-
6	ETH out mac_speed_o0	ETH out mac_speed_o1	ETH out lpi_intr_o	ETH out pmt_intr_o	RCC pwrds_sys	RCC pwrds_mcu	RCC pwrds_mpu	RCC CM4_SLEEPDE EP
7	-	-	-	-	-	-	-	-
8	DDRCTRL csysreq_ddrc	DDRCTRL csysack_ddrc	DDRCTRL cactive_ddrc	DDRCTRL stat_ddrc_reg_ selfref_type[1]	DDRCTRL stat_ddrc_reg_ selfref_type[0]	DDRCTRL dfi_init_complete	DDRCTRL dfi_ctrlupd_req	DDRCTRL lp_req

Table 550. HDP signal multiplexing (continued)

HDP_MUX	HDP7	HDP6	HDP5	HDP4	HDP3	HDP2	HDP1	HDP0
	MUX7[3:0]	**MUX6[3:0]**	**MUX5[3:0]**	**MUX4[3:0]**	**MUX3[3:0]**	**MUX2[3:0]**	**MUX1[3:0]**	**MUX0[3:0]**
9	DDRCTRL hpr_credit_cnt [4:0]==0	DDRCTRL lpr_credit_cnt[4:0]==0	DDRCTRL wr_credit_cnt [4:0]==0	DDRCTRL cactive_0	DDRCTRL cactive_1	DDRCTRL perf_op_is_ refresh	DDRCTRL cactive_ddrc_asr	PWR DDR_RET_ ENABLE_N
10	DTS valobus1_4	DTS valobus1_3	DTS valobus1_2	DTS valobus1_1	DTS valobus1_0	DDRCTRL gskp_dfi_lp_req	-	DTS CLK_PTAT
11	DTS valobus2_4	DTS valobus2_3	DTS valobus2_2	DTS valobus2_1	DTS valobus2_0	-	-	-
12	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-
15	gpoval[7]	gpoval[6]	gpoval[5]	gpoval[4]	gpoval[3]	gpoval[2]	gpoval[1]	gpoval[0]

63.4 HDP registers

63.4.1 HDP control register (HDP_CTRL)

Address offset: 0x000

Reset value: 0x0000 0000

HDP Control

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Enable HDP
Valid if enabled in BSEC.

63.4.2 HDP multiplexers control register (HDP_MUX)

Address offset: 0x004

Reset value: 0x0000 0000

HDP multiplexing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX7[3:0]				MUX6[3:0]				MUX5[3:0]				MUX4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MUX3[3:0]				MUX2[3:0]				MUX1[3:0]				MUX0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **MUX7[3:0]**: Select the HDP7 output among the 16 available signals

Bits 27:24 **MUX6[3:0]**: Select the HDP6 output among the 16 available signals

Bits 23:20 **MUX5[3:0]**: Select the HDP5 output among the 16 available signals

Bits 19:16 **MUX4[3:0]**: Select the HDP4 output among the 16 available signals

Bits 15:12 **MUX3[3:0]**: Select the HDP3 output among the 16 available signals

Bits 11:8 **MUX2[3:0]**: Select the HDP2 output among the 16 available signals

Bits 7:4 **MUX1[3:0]**: Select the HDP1 output among the 16 available signals

Bits 3:0 **MUX0[3:0]**: Select the HDP0 output among the 16 available signals

63.4.3 HDP read back value register (HDP_VAL)

Address offset: 0x010

Reset value: 0x0000 0000

HDP value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPVAL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPVAL[7:0]**: HDP read back value
Provides the value of the HDP signals.

63.4.4 HDP general purpose output set register (HDP_GPOSET)

Address offset: 0x014

Reset value: 0xXXXX XXXX

HDP GPO set

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOSET[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOSET[7:0]**: HDP general purpose output set
When a bit is written to 1, the corresponding HDP GPO is set. Writing a bit to 0 has no effect.

63.4.5 HDP general purpose output clear register (HDP_GPOCLR)

Address offset: 0x018

Reset value: 0xXXXX XXXX

HDP GPO clear

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOCLR[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOCLR[7:0]**: HDP general purpose output clear
When a bit is written to 1, the corresponding HDP GPO is cleared. Writing a bit to 0 has no effect.

63.4.6 HDP general purpose output value register (HDP_GPOVAL)

Address offset: 0x01C

Reset value: 0x0000 0000

HDP GPO value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOVAL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOVAL[7:0]**: HDP general purpose output value
When written, defines the value of the HDP GPO. When read, provides the current GPO value.

63.4.7 HDP version register (HDP_VERR)

Address offset: 0x3F4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV[3:0]				MINREV[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **MAJREV[3:0]**: Major revision of the IP

Bits 3:0 **MINREV[3:0]**: Minor revision of the IP

63.4.8 HDP IP identification register (HDP_IPIDR)

Address offset: 0x3F8

Reset value: 0x0003 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: IP Identifier

63.4.9 HDP size identification register (HDP_SIDR)

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SID[31:0]**: Size identifier

63.4.10 HDP register map

Table 551. HDP register map and reset values

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**HDP_CTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x004	**HDP_MUX**	MUX7[3:0]				MUX6[3:0]				MUX5[3:0]				MUX4[3:0]				MUX3[3:0]				MUX2[3:0]				MUX1[3:0]				MUX0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008 - 0x00C	Reserved																																
0x010	**HDP_VAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPVAL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	**HDP_GPOSET**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOSET[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	**HDP_GPOCLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOCLR[7:0]							
	Reset value																									X	X	X	X	X	X	X	X

Table 551. HDP register map and reset values (continued)

Offset	Register Name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	**HDP_GPOVAL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOVAL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x020 - 0x3F0	Reserved																																
0x3F4	**HDP_VERR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAJREV [3:0]				MINREV [3:0]			
	Reset value																									0	0	0	1	0	0	0	0
0x3F8	**HDP_IPIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x3FC	**HDP_SIDR**	SID[31:0]																															
	Reset value	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	1

Refer to *Section 2.5 on page 153* for the register boundary addresses.

64 Debug support (DBG)

64.1 Introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Independent breakpoint debugging of each CPU core in the system
- Code execution tracing
- Software instrumentation
- Cross-triggering
- The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components:

- SWJ-DP: JTAG/Serial-wire debug port
- AXI-AP: AXI access port
- AHB-AP: AHB access port
- APB-AP: APB access port
- ITM: Instrumentation Trace Macrocell
- DWT: Data Watchpoint and Trace
- ETM: Embedded Trace Macrocell
- ETF: Embedded Trace FIFO
- TPIU: Trace Port Interface Unit
- SWO: Serial Wire Output
- CTI: Cross Trigger Interface
- CTM: Cross Trigger Matrix
- Timestamp Generator
- STM: System Trace Macrocell

These components are described in *Section 64.10* to *Section 64.12*. More information can be found in the Arm® documents referenced in *Section 64.13*.

64.2 Debug use cases

The trace and debug system has been designed to support a variety of typical use cases:

- Low cost software trace (Cortex®-M4 only)

  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt detection and program counter sampling.
- External debugging of each core independently

  All processor cores can be simultaneously and independently debugged using equipment connected to the JTAG/SWD debug port. This allows breakpoint and watchpoint setting, code stepping, memory access etc.
- Self-hosted debugging of each core independently

  All processor cores can be simultaneously and independently debugged by software running on one of the cores. This allows breakpoint and watchpoint setting, code stepping, memory access etc.
- Synchronous debugging of both cores

  When one core stops due to a breakpoint or a debugger stop command, the other core can be stopped as well. Similarly, the cores can be restarted at the same time. This allows debugging of loosely coupled applications which require the processors to remain synchronized.
- Tracing code execution from one or more cores via the trace port

  Trace information from all cores is combined into a single trace stream and output to a trace port analyser in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.
- Code and hardware instrumentation via the trace port

  Software “printf” data from the processors, memory content via DMA, and hardware event trace information can be generated using the system trace macrocell. The information is combined with the processor trace stream and output to a trace port analyzer in real time.
- Capturing trace continuously in a circular buffer

  Instead of streaming it off-chip, the combined trace information can be stored on-chip in a circular buffer. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal, an internal event, etc.
- Draining the buffer to the trace port

  The stored trace can be dumped off-chip to the trace port analyzer. The buffer draining can be initiated by the debugger, software, external trigger, internal event etc.
- Reading the buffer with the debugger

  The debugger can read the contents of the trace buffer via the debug port. This is slower than the trace port, but allows basic trace functionality on the debugger without the cost of a trace port analyzer.
- Self-hosted trace analysis

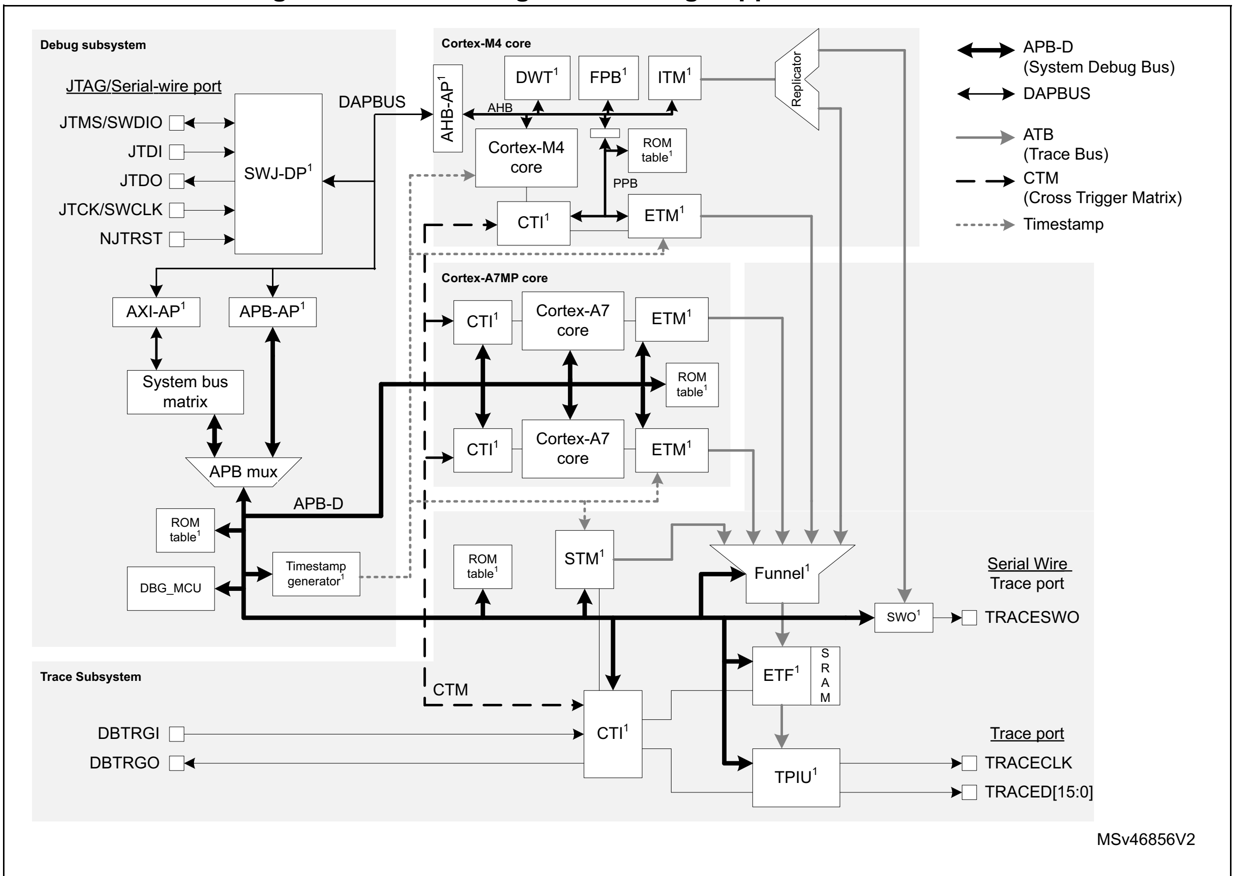
  The trace buffer can be read by any of the processor cores, or transferred into system memory by DMA. This powerful feature allows built-in test software to perform real-time code execution monitoring, fault analysis and identification, autonomous exception handling, and others.

- Uploading stored trace

  The stored trace can also be uploaded to a host machine using one of the MPU's many communications interfaces (such as USB, USART, SPI, I2C, Ethernet, CAN, and others). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.

64.3 DBG block diagram

Figure 783. Block diagram of debug support infrastructure



1. Arm® CoreSight™ component

64.4 DBG pins

Table 552. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	I/O	Serial wire data in/out	Refer to datasheet
JTCK/SWCLK	I	JTAG test clock	I	Serial wire clock	
JTDI	I	JTAG test data input	-	-	
JTDO	O	JTAG test data output	-	-	
NJTRST	I	JTAG test reset	-	-	

Table 553. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED[15:0]	O	Trace synchronous data out (can be 1, 2, 4, 8 or 16 pins)	Refer to datasheet
TRACECLK	O	Trace clock	

Table 554. Serial Wire Trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Trace asynchronous data out	Refer to datasheet[(1)]

1. TRACESWO is multiplexed with JTDO. This means that single wire trace is only available when using the serial wire debug interface, not when using JTAG

Table 555. Trigger pins

Pin name	Type	Description	Pin assignment
DBTRGI	I	External trigger input to CTI	Refer to datasheet
DBTRGO	O	External trigger output from CTI	
DBTRGIO	I/O	External Trigger Bi-directional[(1)]	

1. DBTRGIO can be configured as an input or an output by the TRGOEN bit in the DBGMCU_CR register. If configured as an input, it is connected to DBTRGI. If an output, it is connected to DBTRGO. This is because DBTRGI and DBTRGO are not available on certain packages.

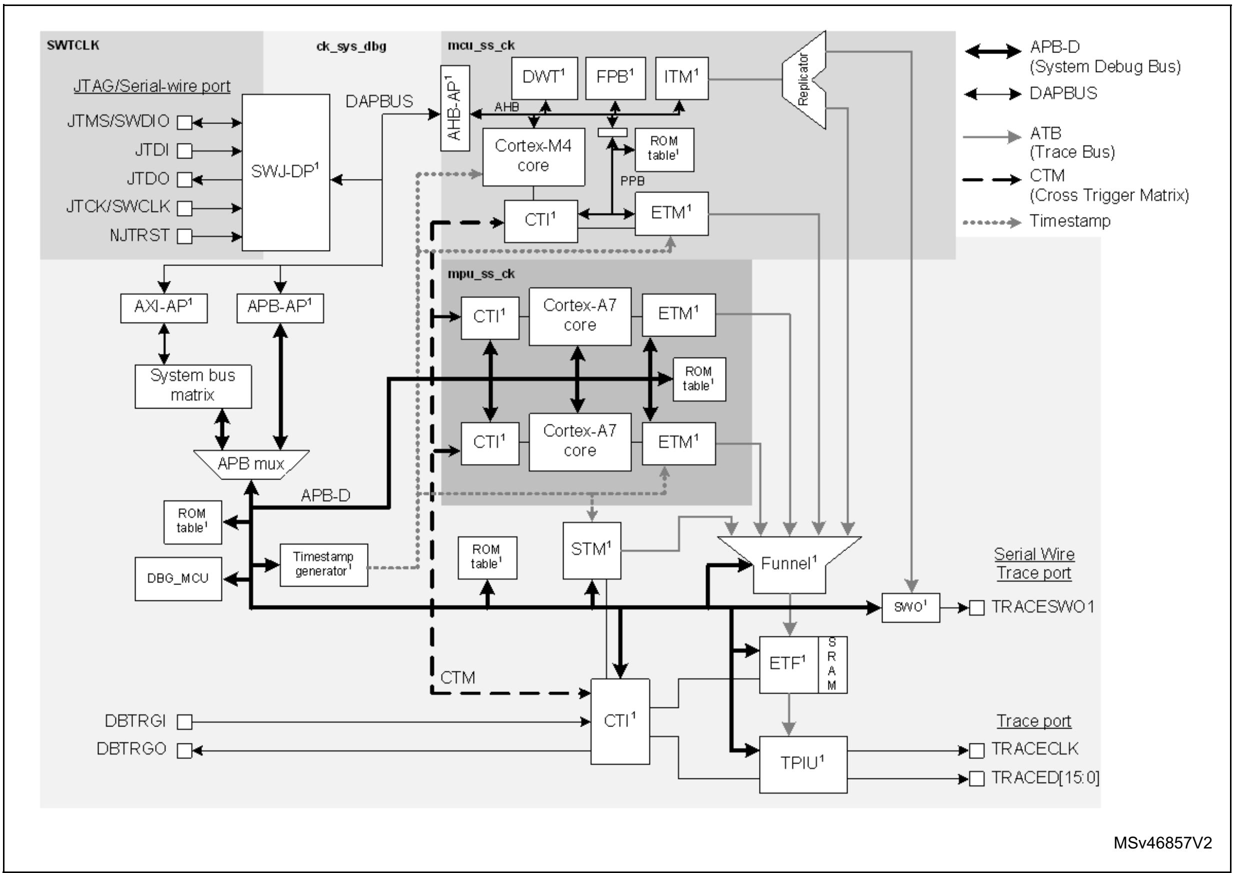
64.5 DBG power and clocking

64.5.1 DBG power domains

All debug components including the SWJ-DP are in the core power domain. This means that the debugger is not able to wake the device from Standby mode.

64.5.2 DBG clocks

Figure 784. Debug clock domains



1. Arm® CoreSight™ component

The debugger supplies the clock for the debug port, SWTCLK, via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the system debug clock domain, ck_sys_dbg. The remainder of the debug subsystem, as well as the trace subsystem, is in this domain. ck_sys_dbg is a gated version of the AXI bus clock, aclk.

The clock is enabled by the debugger using the CDBGPWRUPREQ bit in the debug port CTRL/STAT register. It can also be enabled by software writing bit DBGCKEN in RCC debug configuration register (RCC_DBGCFGR). The clock must be enabled before the debugger or software can access any of the debug features on the device. It should be disabled when debug and trace are not used to avoid wasting energy.

The APB-D (debug APB bus) clock (ck_apb_dbg) runs at half of the ck_sys_dbg frequency. Some of the CoreSight™ components on this bus operate at the APB-D frequency, notably the timestamp generator (TSGEN).

The trace port (TPIU) output clock, ck_trace, is generated using TRACEDIV programmable divider in RCC debug configuration register (RCC_DBGCFGR). Its maximum frequency is half that of ck_sys_dbg. The trace_ck signal is output on the TRACECLK GPIO pin when it is configured for trace function. The same clock is used to drive the serial wire output (SWO).

Note: TRACECLK is enabled by setting TRACECKEN to 1 in RCC debug configuration register (RCC_DBGCFGR). It should be activated whenever tracing is required, even if neither the TPIU nor the SWO are active (that is when using the embedded trace FIFO as a software buffer). This is to avoid filling up the SWO FIFO which would then block the trace output from the ITM in the Cortex®-M4.

The debug and trace components included in the processors (ETM, ITM, DWG, FPB, CTI etc) are clocked with the corresponding core clock (mpu_ss_ck for the Cortex®-A7 subsystem, mcu_ss for the Cortex®-M4 subsystem).

64.5.3 Debug and low-power modes

The STM32MP153 device includes power saving features which allow the various domain clocks and even the core power to be switched off when not required. If the core power is switched off (Standby mode), or the system debug domain is not clocked (Stop mode), all debug components will be inaccessible to the debugger. To avoid this, power saving mode emulation has been implemented. If emulation is enabled, the device still enters the programmed low power mode, but its clock and power are maintained. In other words, the domain behaves as if it is in low power mode, but the debugger does not lose the connection.

Emulation mode is programmed in the MCU Debug (DBGMCU) unit. For more information, refer to *Section 64.10.8*.

If the device is in Stop mode (ie. all clocks off) without emulation and the debugger wishes to wake the system up, it can do so by asserting the CDBGPWRUPREQ bit in the debug port CTRL/STAT register. This restarts the clocks (and switches on the high power regulator if needed) but does not generate any interrupt or event to either processor core. When the wake-up has completed, the CDBGPWRUPACK bit in the same register will go high. In this state the debugger can access the memories and peripheral registers, and trigger a wakeup of the CPUs if required.

64.5.4 DBG reset

The debug components, except for the debug port, are reset by their respective clock domain resets. The debug port (SWJ-DP) is reset by the nTRST or vcore_rst resets. The access ports can be reset under debugger control, using the CDBGRSTREQ/ACK bits in the CTRL/STAT register of the debug port.

The debugger sets the CDBGRSTREQ bit to request a reset. It then polls the CDBGRSTACK bit until it goes high, indicating that the reset has completed, following which it deasserts the CDBGRSTREQ bit.

64.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features can not be used to alter or compromise the normal operation of the finished product, these features can be disabled, or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of non-secure code.

64.6.1 Authentication signals

There are six authentication signals used by the system to determine what features are enabled or disabled. These signals controlled in the BSEC debug configuration (BSEC_DENABLE) are:

- **deviceen**: controls access to debug components via the external debug port.
  0: external debugger has no access to debug components. Debugger access to system interconnect is not affected.
  1: external debugger has access to debug components
- **dbgen**: global enable for all debug and trace features
  0: all debug features are disabled. External and software access to debug components is still possible. External access to system interconnect is disabled.
  1: debug features in non-secure mode are enabled. External access to non-secure system interconnect is enabled. Debug features in secure mode and external access to secure system interconnect are dependent on the state of the **spiden** signal.
- **spiden**: enables debug in secure privileged mode when **dbgen** = 1
  0: debug features are disabled in secure privileged mode. External access to secure system interconnect is disabled.
  1: debug features are enabled in secure privileged mode. External access to secure system interconnect is enabled.
- **niden**: enables trace and performance monitoring (non-invasive debug)
  0: trace generation is disabled.
  1: trace generation in non-secure mode is enabled. Trace generation in secure privileged mode is dependent on the state of the **spniden** signal.
- **spniden**: enables trace and performance monitoring in secure privileged mode when **niden** = 1.
  0: trace generation is disabled in secure privileged mode.
  1: trace generation is enabled in secure privileged mode.
- **dbgswenable**: enables self-hosted debug
  0: software access to all debug components is disabled
  1: software access to all debug components is enabled

Table 556 lists which signals are connected to which debug and trace components.

Table 556. Authentication signal connectivity

	deviceen	dbgen	spiden	niden	spniden	dbgswenable
Debug APB-AP	X	-	-	-	-	X
Cortex®-M4 AHB-AP	X(1)	X	-	-	-	-
System bus AXI-AP	-	X	X	-	-	-
Cortex®-M4 DWT/FPB/SCS/ITM	-	X	-	-	-	-
Cortex®-M4 ETM	-	-	-	X	-	-
Cortex®-M4 CTI	-	X	-	X	-	-
Cortex®-A7 MPCore Debug unit	-	X	X	X	X	X
Cortex®-A7 MPCore ETM	-	X	-	X	-	-

Table 556. Authentication signal connectivity (continued)

	deviceen	dbgen	spiden	niden	spniden	dbgswenable
Cortex®-A7 MPCore CTI	-	X	-	X	-	-
STM	-	X	X	X	X	-

1. **deviceen** connects to DAPEN in the Cortex®-M4

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter, or to the relevant Arm® technical documentation.

The initial state of the signals are set at power-on reset, according to *Table 557*. The state of the signals can be modified in the BSEC debug configuration (BSEC_DENABLE) register after boot-up by software with secure privileged access rights.

Table 557. Authentication signal initial states

Device state (set by OTP fuse)	Authentication signal default state	Description
Development device	All ones	All debug and trace is enabled at boot-up.
Production device	All zeros	All debug and trace is disabled at boot-up. Secure software should enable debug as required after booting.

64.7 Chip level TAP controller (CLTAPC)

The JTAG interface is connected to the internal scan chain of the IC. It is used for testing as well as for debug. The first component in the scan chain is the Chip Level TAP Controller which implements a TAP state machine based on IEEE Std 1149.1-1990. It allows access to the built-in manufacturing test features.

The CLTAPC must be bypassed in order to access the debug support functions provided by the SWJ-DP. This is done by prefixing the SWJ-DP instructions with the CLTAPC bypass instruction sequence: 11111.

The CLTAPC also contains three 16-bit registers which can be accessed using the CTRL instruction sequence: 01000, followed by 1111 to bypass the SWJ-DP. Next, the register address is shifted into the DR (LSB first). Finally, the 16-bit register content is shifted into or out of the DR, according to whether the register is read or write.

The CLTAPC registers are listed in *Table 558*.

Table 558. CLTAPC registers

Name	Address	Description	Read/Write
SECSTS	001	Status of the BSEC_DENABLE register	R
DBGCTL	010	Connected to BSEC_JTAGIN register	W
DBGSTS	011	Connected to BSEC_JTAGOUT register	R

For details of the BSEC register bits, refer to *Section 3: Boot and security and OTP control (BSEC)*.

64.8 Serial wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight™ component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- A 5-pin standard JTAG interface (JTAG-DP)
- A 2-pin (clock + data) “serial-wire debug” port (SW-DP)

The two modes are mutually exclusive, since they share the same I/O pins.

By default the JTAG-DP is selected after a system or power-on reset. The five I/O pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and NJTRST, as well as a pull-down resistor on JTCK/SWCLK.

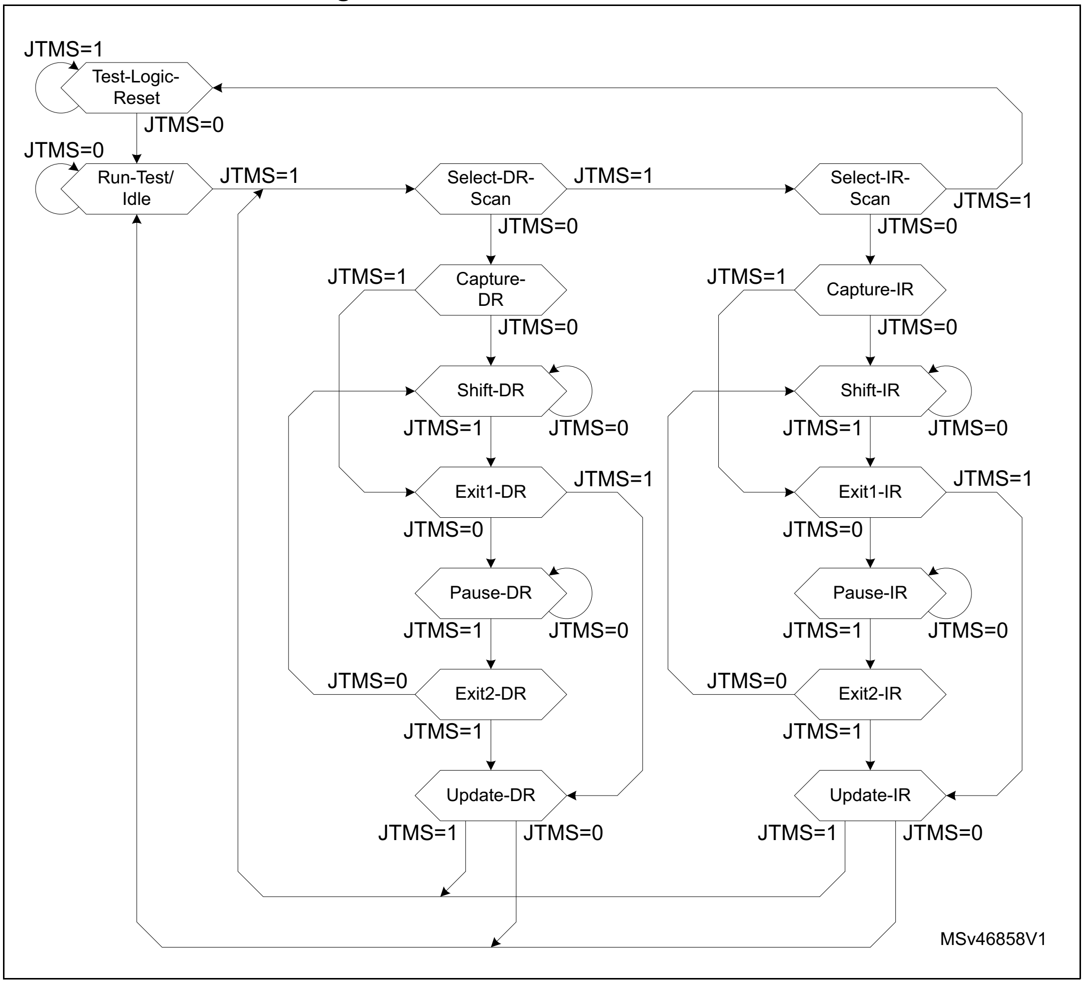
A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

...(50 or more ones)...,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,...(50 or more ones)...

JTCK/SWCLK must be cycled for each data bit.

64.8.1 JTAG debug port

Figure 785. JTAG TAP state machine



The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE Std 1149.1-1990. The state machine is shown in *Figure 785*. The state machine controls two scan chains, one associated with an instruction register (IR) and one with a number of data registers (DR).

The operation of the JTAG-DP is as follows:

When the TAPSM goes through the Capture-IR state, 0001 is transferred onto the Instruction Register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:

- The LSB of the IR scan chain is output on JTDO.
- Bit[n] of the IR scan chain is transferred to bit[n-1].
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred into the Instruction Register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one of the Data Registers onto one of the DR scan chains, connected between JTDI and JTDO.

The value held in the Instruction Register determines which Data Register, and associated DR scan chain, is selected.

This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shift in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred into the selected Data Register.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

The NJTRST signal when active resets the state machine asynchronously to the Test-Logic-Reset state.

The Data registers corresponding to the 4-bit IR instructions are listed in *Table 559*.

Table 559. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – Bits 31:1 = Reserved – Bit 0 = APABORT: write 1 to generate a AP abort.
1001	(BYPASS)	1	Reserved: BYPASS selected

Table 559. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1010	DPACC	35	Debug port access register Initiates the debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT Others = Reserved
1011	APACC	35	Access port access register Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register. Bit 0 = RnW= Read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit Acknowledge: 010 = OK/FAULT 001 = WAIT Others = Reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	ID Code 0x6BA0 0477: Arm® JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO

The DR registers are described in more detail in the *Arm® Debug Interface Architecture Specification* [1].

64.8.2 Serial Wire debug port

The Serial Wire Debug protocol uses the two pins:

- SWCLK: clock from host to target
- SWDIO: bi-directional serial data (100 kOhm pull-up required)

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three phases:

1. Packet request (8 bits) transmitted by the host
2. Acknowledge response (3 bits) transmitted by the target.
3. Data transfer (33 bits) transmitted by the host (in the case of a write) or target (in the case of a read).

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock cycle turn-around time is inserted.

Table 560. Packet request

Bit field	Name	Description
0	Start	Must be 1
1	APnDP	0: DP register access - see *Table 563* for a list of DP registers 1: AP register access - see *Section 64.9*
2	RnW	0: write request 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 563*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host. Must be read as 1 by target.

Table 561. ACK response

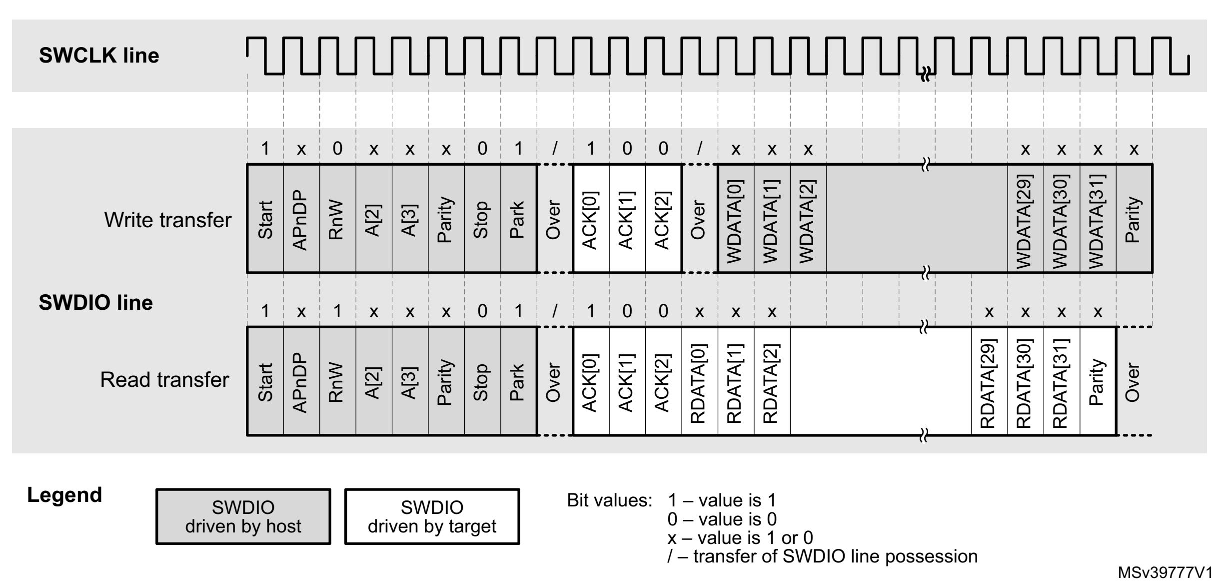
Bit field	Name	Description
2:0	ACK	000: FAULT 010: WAIT 100: OK

Table 562. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single bit parity of 32 data bits

Figure 786 shows examples of successful write and read transfers.

Figure 786. SWD successful data transfer



In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is cancelled, unless overrun detection is enabled, in which case the data will be ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists of 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the *Arm® Debug Interface Architecture Specification [1]*.

Note: The SWJ-DP implements SWD protocol version 2

64.8.3 Debug port registers

The SW-DP and JTAG-DP both access the debug port (DP) registers. These are listed in *Table 563*.

The debugger can access the DP registers as follows:

1. Program the DPBANKSEL field in the DP_SELECT data port register to select the register bank to be accessed (see *Table 563*).
2. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank.
3. Program the RnW bit to select a Read or Write.
4. In the case of a write, program the DATA field with the write data.

If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see *Table 560*). The write data is sent in the data phase.

Table 563. Debug port registers

Address	A(3:2) value	R/W	Description
0x0	00	R	DP_DPIDR register. Contains the IDCODE for the debug port
		W	DP_ABORT register[1]. Aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTAT register.
0x4	01	R/W	If field DPBANKSEL of register DP_SELECT is 0: DP_CTRLSTAT register. Controls the DP and provides status information.
			If field DPBANKSEL of register DP_SELECT is 1: DP_DLCR register[2]. Controls the operating mode of the SWD data link.
			If field DPBANKSEL of register DP_SELECT is 2: DP_TARGETID register. Provides target identification information.
			If field DPBANKSEL of register DP_SELECT is 3: DP_DLPIDR register[3]. Provides the SWD protocol version.
0x8	10	R	DP_RESEND register[4]. Returns the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.
		W	DP_SELECT register. Selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	DP_RDBUFF register: Via JTAG-DP - Used to allow the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation) Via SW-DP - Contains the result of the preceding AP read access, allowing a new AP access to be avoided.
		W	DP_TARGETSEL register[5]. On a write to DP_TARGETSEL immediately following a line reset sequence, the target is selected if both the following conditions are met: Bits [31:28] match bits [31:28] in the DP_DLPIDR. Bits [27:0] match bits [27:0] in the DP_TARGETID register. Writing any other value de-selects the target. Debug tools must write 0xFFFF FFFF to de-select all targets. This is an invalid TARGETID value. All other invalid TARGETID values are reserved.

1. Access to the AP DP_ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “Reserved” via JTAG-DP.
3. Only accessible via SW-DP. Register is “Reserved” via JTAG-DP.
4. Only accessible via SW-DP. Register is “Reserved” via JTAG-DP.
5. Only accessible via SW-DP. Register is “Reserved” via JTAG-DP.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: revision code
0x6

Bits 27:20 **PARTNO[7:0]**: debug port part number
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN:** minimal debug port implementation
0x0: minimal debug port (MINDP) not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm®

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORT)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	STKCMPCLR	DAPABORT
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: overrrun error clear
0: no effect
1: clears bit STICKYORUN of register DP_CTRLSTAT

Bit 3 **WDERRCLR**: write data error clear
0: no effect
1: clears bit WDATAERR of register DP_CTRLSTAT

Bit 2 **STKERRCLR**: sticky error clear
0: no effect
1: clears bit STICKYERR of register DP_CTRLSTAT

Bit 1 **STKCMPCLR**: sticky compare clear
0: no effect
1: clears bit STICKYCMP of register DP_CTRLSTAT

Bit 0 **DAPABORT**: current AP transaction abort
If an excessive number of WAIT responses are returned, indicating that the transaction has stalled, the current AP transaction is aborted.
0: no effect
1: aborts transaction

Debug port control/status register (DP_CTRLSTAT)

Address offset: 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	CDBGRSTACK	CDBGRSTREQ	Res.	Res.	TRNCNT[11:4]							
		r	rw	r	rw			rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNCNT[3:0]				MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: debug power up acknowledge
Status of the debug domain power up acknowledge signal from the power controller.
0: domain powered down
1: domain powered up

Bit 28 **CDBGPWRUPREQ**: debug power up request
Controls the debug domain power up request signal to the power controller.
0: request power down
1: request power up

Bit 27 **CDBGRSTACK**: debug reset acknowledge
Status of the debug domain reset acknowledge signal from the reset controller.
0: domain reset inactive/pending
1: domain reset complete

Bit 26 **CDBGRSTREQ**: debug reset request
Controls the debug domain reset request signal to the reset controller.
0: de-assert reset request/request inactive
1: assert reset request/request active

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]**: transaction counter
To program a sequence of transactions to incremental addresses via an AP, TRNCNT is loaded with the number of transactions to perform. It is decremented at the successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: lane comparison mask
Indicates the bytes to be masked in pushed-compare and pushed-verify operations (when bit TRNMODE of register DP_CTRLSTAT is 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.
1XXX: includes byte lane 3 in comparisons
X1XX: includes byte lane 2 in comparisons
XX1X: includes byte lane 1 in comparisons
XXX1: includes byte lane 0 in comparisons

Bit 7 **WDATAERR**: write data error in SW-DP
Indicates any of the following cases:
– there is a a parity or framing error on the data phase of a write
– a write that has been accepted by the DP is then discarded without being submitted to the AP
This bit is reset by writing 1 to the WDERRCLR bit of the DP_ABORT register.
0: no error
1: error has occurred
Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response in SW-DP
Indicates the response to the last AP read access.
0: read not OK
1: read OK
Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: transaction error(1)

Indicates that an error occurred in an AP transaction.

0: no error

1: error has occurred

In the SW-DP, this bit is reset by writing 1 to the STKERRCLR bit of the DP_ABORT register. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 4 **STICKYCMP**: compare match(1)

Indicates that a match occurred in a pushed operation.

0: match if TRNMODE = 0x1; no match if TRNMODE = 0x2

1: no match if TRNMODE = 0x1; match if TRNMODE = 0x2

In the SW-DP, this bit is reset by writing 1 to the STKCMPCLR bit of the DP_ABORT register. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bits 3:2 **TRNMODE[1:0]**: transfer mode for AP write operations

For read operations, this field must be set to 0.

0: normal operation. AP transactions are passed directly to the AP.

1: pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they do not match, the STICKYCMP bit is set.

2: pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data and if they match, the STICKYCMP bit is set.

3: reserved

In pushed operation, only the data bytes indicated by the MASKLANE field are included in the compare.

Bit 1 **STICKYORUN**: overrun(1)

Indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.

0: no overrun

1: overrun occurred

In the SW-DP, this bit is reset by writing 1 to the ORUNERRCLR bit of the DP_ABORT register. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT**: overrun detection mode enable

0: overrun detection disabled

1: overrun detection enabled. In the event of an overrun, the STICKYORUN bit will be set and subsequent transactions will be blocked until the STICKYORUN bit is cleared.

1. Access type is ‘r’ in SW-DP and ‘rw’ in JTAG-DP.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: tristate period for SWDIO
0x0: 1 data bit period
0x1: 2 data bit periods
0x2: 3 data bit periods
0x3: 4 data bit periods

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETID)

Address offset: 0x4

Reset value: 0x2500 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: target revision
0x2: Rev. B

Bits 27:12 **TPARTNO[15:0]**: target part number
0x5000: STM32MP15x

Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: target instance number
Defines the instance number for this device in a multi-drop system.
0x0: instance 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial Wire Debug protocol version
0x1: version 2

DP resend register (DP_RESEND)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: last AP read or DP_RDBUFF read value
Returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.

DP access port select register (DP_SELECT)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: access port select
Selects the access port for the next transaction.
0x0: AP0 - System bus access port (AXI-AP)
0x1: AP1 - Debug bus access port (APB-AP)
0x2: AP2 - Cortex®-M4 debug access port (AHB-AP)

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select
Selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select
Selects the register at address 0x4 of the debug port.
0x0: DP_CTRLSTAT register
0x1: DP_DLCR register
0x2: DP_TARGETID register
0x3: DP_DLPIDR register
Others: reserved

DP read buffer register (DP_RDBUFF)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: last AP read access value
Contains the value that was returned by the last AP read access. The value returned by an AP read access can either be obtained using a second read access to the same address, which will initiate a new transaction on the corresponding bus, or else it can be read from this register, in which case no new AP transaction occurs.

DP target identification register (DP_TARGETSEL)

Address offset: 0xC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				TPARTNO[15:4]											
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

Bits 31:28 **TINSTANCE[3:0]**: target instance number

Defines the instance number for the target device in a multi-drop system. These bits must be written with the same value as field TINSTANCE in register DP_DLPIDR in order to select this device.

Bits 27:12 **TPARTNO[15:0]**: target part number

Defines the part number for the target device. These bits must be written with the same value as field TPARTNO in register DP_TARGETID in order to select this device.

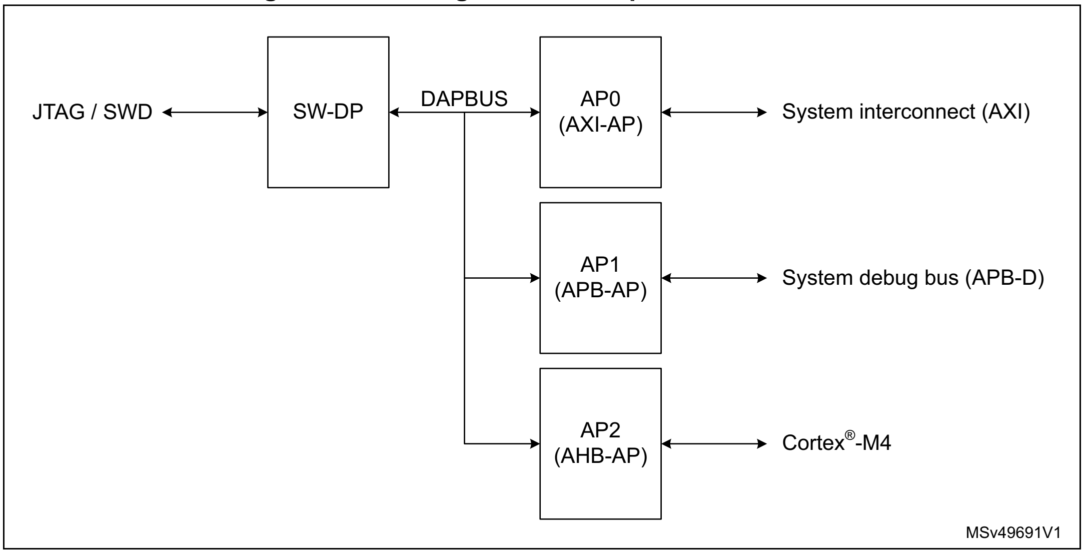
Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code

Defines the JEDEC code for the target device. These bits must be written with the same value as field TDESIGNER in register DP_TARGETID in order to select this device.

Bit 0 Reserved, must be kept at reset value.

64.9 Access ports

Figure 787. Debug and access port connections



There are three access ports (AP) attached to the DP:

- AP0: System bus access port (AXI-AP). Allows access to the system bus matrix. This gives visibility of all the memory and peripherals accessible via the AXI and AHB interconnects, except those on the system APB debug bus. No CoreSight™ components are accessible via this port.
- AP1: Debug access port (APB-AP). Allows access to the CoreSight™ components on the debug APB bus, including the Cortex®-A7. The DBGMCU is also located on this bus.
- AP2: Cortex®-M4 access port (AHB-AP). Allows access to the debug and trace features integrated in the Cortex®-M4 processor core via its PPB bus.

All three access ports are of type MEM-AP, that is to say the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organised in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 567*.

The address of the AP registers is composed of

- Bits [7:4]: contents of the DP_SELECT register APBANKSEL field in the DP (refer to *DP access port select register (DP_SELECT)*)
- Bits [3:2]: contents of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 559*) or of the SW-DP Packet Request (see *Table 560*), depending on the debug interface used
- Bits [1:0]: Always set to 0

The contents of the DP_SELECT register APSEL field in the DP define which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

1. Program the DP_SELECT register APSEL field in the DP to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed (refer to *DP access port select register (DP_SELECT)*).
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a Read or Write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see *Table 560*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (that is using the above AP register access procedure) as follows:

1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to Banked Data register BDn will trigger an access to address TAR[31:4] + n (this allows up to four consecutive addresses to be accessed without changing the address in the TAR register).

For more detailed information on the MEM-AP, refer to the *Arm® Debug Interface Architecture Specification [1]*.

64.9.1 Authentication

The effect of the authentication signals for each access port is shown in *Table 564* to *Table 566*.

Table 564. AP0 (system interconnect) authentication behavior

dbgen	spiden	Others	Behavior
0	X	X	All transfers blocked. Any attempt to perform a transfer will result in an error.
1	0	X	Secure transfers blocked. Any attempt to perform a secure transfer will result in an error. Non-secure transfers are enabled.
1	1	X	Both non-secure and secure transfers are enabled.

Table 565. AP1 (Debug APB) authentication behavior

deviceen	Others	Behavior
0	X	All transfers blocked.
1		Transfers enabled. Note: access to certain components may return an error depending on the state of the other authentication signals.

Table 566. AP2 (Cortex®-M4) authentication behavior

deviceen(1)	dbgen	Others	Behavior
0	X	X	All transfers blocked. Any attempt to perform a transfer will result in an error.
X	0		All transfers blocked. Any attempt to perform a transfer will result in an error.
1	1		Transfers enabled.

1. **deviceen** is connected to the DAPEN input of the Cortex®-M4

64.9.2 MEM-AP registers

Table 567. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	CSW	Control/Status Word register - Refer to section *AP0 control/status word register (AP0_CSW)* *AP1 control/status word register (AP1_CSW)* *AP2 control/status word register (AP2_CSW)*
0x04	0x0	1	TAR	Transfer Address register - Refer to section *AP transfer address register (AP_TAR)*
0x08	-	-	-	Reserved
0x0C	0x0	3	DRW	Data Read/Write register - Refer to section *AP data read/write register (AP_DRW)*
0x10	0x1	0	BD0	Banked Data 0 register - Refer to section *AP banked data register 0 (AP_BD0)*
0x14	0x1	1	BD1	Banked Data 1 register - Refer to section *AP banked data register 1 (AP_BD1)*
0x18	0x1	2	BD2	Banked Data 2 register - Refer to section *AP banked data register 2 (AP_BD2)*
0x1C	0x1	3	BD3	Banked Data 3 register - Refer to section *AP banked data register 3 (AP_BD3)*
0x20	0x2	0	ACE	ACE barrier transaction register (AXI-AP only) - Refer to section *AP ACE barrier transaction register (AP_ACE)*
0x24-0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	0xF	1	CFG	Configuration register (AXI-AP only) - Refer to section *AP configuration register (AP_CFG)*

Table 567. MEM-AP registers (continued)

Address	APBANKSEL	A(3:2)	Name	Description
0xF8	0xF	2	BASE	Debug Base Address register (RO) - Refer to section *AP x base address register (APx_BASE) (x = 0 to 2)*
0xFC	0xF	3	IDR	Identification register (RO) - Refer to section *AP x identification register (APx_IDR) (x = 0 to 2)*

AP0 control/status word register (AP0_CSW)

Address offset: 0x0

Reset value: 0x3080 6042

For a detailed description of each field, refer to the *Arm AMBA protocol specifications for the corresponding bus [10.,11.,12.]*.

AP0 (AXI-AP):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PROT[2:0]			CACHE[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DOMAIN[1:0]		ACEENABLE	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PROT[2:0]**: bus transaction protection attributes

This field sets the AxPROT[2:0] protection attributes of the bus transaction.
XX0: unprivileged
XX1: privileged
X0X: secure
X1X: non-secure
0XX: data
1XX: instruction

Bits 27:24 **CACHE[3:0]**: bus transaction cache attributes

This field sets the AxCACHE[3:0] cache attributes of the bus transaction.
XXX0: non-bufferable
XXX1: bufferable
XX0X: non-modifiable (non-cacheable)
XX1X: modifiable (cacheable)
X0XX: no read-allocate
X1XX: read-allocate
0XXX: no write-allocate
1XXX: write-allocate

Bit 23 **SPISTATUS**: status of **spiden** authentication signal (read only)
This signal determines whether the debugger can access secure memory.
0: secure AXI transfers are blocked
1: secure AXI transfers are allowed

Bits 22:15 Reserved, must be kept at reset value.

Bits 14:13 **DOMAIN[1:0]**: shareable transaction attributes
This field defines the shareable attributes for an ACE transaction.
0x0: non-shareable
0x1: shareable, inner domain, includes additional masters
0x2: shareable, outer domain, also includes inner or additional masters
0x3: shareable, system domain, all masters included

Bit 12 **ACEENABLE**: ACE transactions enable, including barriers
0: disable
1: enable

Bits 11:8 **MODE[3:0]**: mode of operation
0x0: normal download or upload
0x1: barrier transaction
0x2-0xF: reserved

Bit 7 **TRINPROG**: transfer in progress (read only)
Indicates if a bus transfer is in progress on the AP.
0: no transfer in progress
1: bus transfer in progress

Bit 6 **DBGSTATUS**: status of DBGEN authentication signal (read only)
This signal determines whether the AXI bus can be accessed.
0: AXI transactions are blocked
1: AXI transactions are allowed

Bits 5:4 **ADDRINC[1:0]**: auto-increment and packing mode on RW data access
Only increments if the current transaction completes without an error response and the transaction is not aborted.
Auto address incrementing and packed data transfers are not performed on access to banked data registers 0x10-0x1C. The value of these bits is ignored in these cases.
If increment is enabled, the address wraps at 1-Kbyte address boundaries.
0x0: no auto-increment
0x1: address is incremented by the size in bytes of the transaction (SIZE field)
0x2: packed transfers enabled. A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transactions corresponding to the programmed transaction size. The data are packed or unpacked accordingly.
0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x0: 8 bits
0x1: 16 bits
0x2: 32 bits
0x3: 64 bits
Others: reserved

AP1 control/status word register (AP1_CSW)

Address offset: 0x0

Reset value: 0x8000 0042

For a detailed description of each field, refer to the *Arm AMBA protocol specifications for the corresponding bus [10.,11.,12.]*.

AP1 (APB-AP):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGSWENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bit 31 **DBGSWENABLE**: software access enable

Defines whether the system debug APB can be accessed by software. This bit is anded with the **dbgswenable** signal from the BSEC.
0: disable software access
1: enable software access if **dbgswenable** signal from BSEC is also high

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: barrier support enabled

Defines if memory barrier operation is supported.
0x0: not supported

Bit 7 **TRINPROG**: transfer in progress (read only)

Indicates if a bus transfer is in progress on the AP.
0x0: no transfer in progress
0x1: bus transfer in progress

Bit 6 **DEVICEEN**: device enabled (read only)

Defines whether the system debug APB can be accessed by the debugger. This bit reflects the state of the **deviceen** signal from the BSEC.
0x0: APB access disabled. Any attempt to initiate a transaction on the AP will return an error.
0x1: APB access enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode

Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address is incremented by the size in bytes of the transaction (SIZE field)
Others: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory transaction access (read only)

0x2: word (32 bits)

AP2 control/status word register (AP2_CSW)

Address offset: 0x0

Reset value: 0x2300 0040

For a detailed description of each field, refer to the *Arm AMBA protocol specifications for the corresponding bus [10.,11.,12.]*.

AP2 (AHB-AP):

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MASTERTYPE	Res.	Res.	Res.	HPROT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r				rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **MASTERTYPE**: type of master (read only)
1: debug

Bits 28:26 Reserved, must be kept at reset value.

Bit 25 **HPROT1**: defines the HPROT(1) attribute of the transaction
0: user access
1: privileged access

Bits 24:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: mode of operation
0x0: Normal download or upload
0x1-0xF: Reserved

Bit 7 **TRINPROG**: transfer in progress (read only)
Indicates if a bus transfer is in progress on the AP.
0x0: no transfer in progress
0x1: bus transfer in progress

Bit 6 **DBGSTATUS**: status of the DBGEN authentication signal (read only)
This signal determines whether the AHB bus can be accessed.
0: AHB transactions are blocked
1: AHB transactions are allowed

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether address is automatically incremented after a transaction. If increment is enabled, the address wraps at 4- Kbyte address boundaries.
Auto address incrementing and packed data transfers are not performed on access to banked data registers 0x10-0x1C. The value of these bits is ignored in these cases.
0x0: no auto-increment
0x1: address is incremented by the size in bytes of the transaction (SIZE field)
0x2: packed transfers enabled. A 32-bit AP access gives rise to 1 x 32-bit, 2 x 16-bit, or 4 x 8-bit bus transactions corresponding to the programmed transaction size. The data is packed or unpacked accordingly.
0x3: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x0: byte (8 bits)
0x1: halfword (16 bits)
0x2: word (32 bits)
Others: reserved

AP transfer address register (AP_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: address of the current transfer (bits 31:0)

AP data read/write register (AP_DRW)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer

In write mode, this is the write data value. In read mode, this is the read data value.
For 64-bit access (AXI-AP only), multiple accesses must be initiated to DRW to make a single AXI access.

Read: the first read of DRW in a sequence initiates a memory access. The first read returns the lower 32 bits of data. Subsequent read access returns the upper 32 bits of data. If a write to the CSW or TAR is initiated before the sequence completes, then the read access is terminated, and read data is no longer available.

Write: the first write to DRW specifies the least significant 32 bits of data to be written. Subsequent write access specifies the most significant 32 bits to be written. If a write to the CSW is initiated before the sequence completes, then the write access is not initiated on the AXI interface.

Combining partial reads and writes in a sequence terminates the earlier access. Also, the latest access is not recognized.
Any write access to the CSW register, TAR, or to any other register in the AP during a sequence terminates the ongoing access. Also, the current access is not recognized.
If a write sequence is terminated, then there is no write on the AXI interface.

AP banked data register 0 (AP_BD0)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0x0.

AP banked data register 1 (AP_BD1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0x4.

AP banked data register 2 (AP_BD2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0x8.

AP banked data register 3 (AP_BD3)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0xC.

AP ACE barrier transaction register (AP_ACE)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BARTRAN[1:0]		TRGBARTRAN
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:1 **BARTRAN[1:0]**: barrier transactions
0x0: barrier with normal access
0x1: memory barrier
0x2: reserved
0x3: synchronization barrier

Bit 0 **TRGBARTRAN**: enable barrier transaction
0: disable
1: enable

AP configuration register (AP_CFG)

Address offset: 0xF4

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	Res.	LA
													r		r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LD**: large data
Indicates support for data items larger than 32 bits (AXI-AP only).
1: 64-bit data supported

Bit 1 Reserved, must be kept at reset value.

Bit 0 **LA**: long address
Indicates support for greater than 32-bit addressing.
0: 32 or fewer bits of address

AP x base address register (APx_BASE) (x = 0 to 2)

Address offset: 0xF8

Reset value: Block 0: 0x0000 0002

Reset value: Block 1: 0xE008 0003

Reset value: Block 2: 0xE00F F003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]**: base address

Corresponds to bits 31 to 12 of the ROM table address for the AP. The 12 LSBs are zeros since the ROM table must be aligned on a 4-Kbyte boundary.
AP0 (AXI system bus): N/A (no ROM table)
AP1 (system debug APB): 0xE0080
AP2 (Cortex®-M4): 0xE00FF

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: base address register format

1: Arm® debug interface v5

Bit 0 **ENTRYPRESENT**: debug component presence

Indicates that debug components are present on the access port bus.
0: no debug components present
1: debug components are present

AP x identification register (APx_IDR) (x = 0 to 2)

Address offset: 0xFC

Reset value: Block 0: 0x3477 0004

Reset value: Block 1: 0x4477 0003

Reset value: Block 2: 0x2477 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: revision code
0x2: r0p3
0x3: r1p0
0x4: r0p5
0x6: r0p7

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm®

Bit 16 **MEMAP**: memory access port
0x1: standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type
0x04: AXI-AP
0x03: APB-AP
0x11: Cortex®-M4 AP

64.10 System debug features

The system debug features include the following CoreSight™ components:

- System ROM tables
- Global trace timestamp generator (TSGEN)
- System cross trigger interface (CTI)
- Cross trigger matrix (CTM)
- Trace port interface unit (TPIU)
- Trace bus funnel (CSTF)
- Embedded trace FIFO (ETF)
- Serial Wire Output (SWO)

These components are accessible by the debugger via the system APB-AP and its associated APB debug bus. They are also accessible by the Cortex®-A7 and Cortex®-M4 processors.

The MCU debug unit (DBG_MCU) is also accessed via the Debug APB. This non-CoreSight™ component contains registers for configuring the behavior of the device in debug mode.

64.10.1 System ROM tables

There are two ROM tables on the Debug APB bus. The ROM table is a CoreSight™ component that contains the base addresses of the CoreSight™ debug components on the APB bus. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

The first table points to the CoreSight™ components located in the debug subsystem: SWO, TSG, and the Cortex®-A7 debug components, as well as to the Trace subsystem ROM table on the debug APB. The DBG_MCU is not referenced by the table as it is not a standard CoreSight™ component. The table occupies a 4-Kbyte, 32-bit wide chunk of APB address space, from 0xE008 0000 to 0xE008 0FFC when accessed by the debugger, and 0x5008 0000 to 0x5008 0FFC when accessed from the system bus.

Table 568. Debug subsystem ROM table

Address offset in ROM table	Component name	Component base address (debugger)	Component base address (system bus)	Component address offset	Size	Data
0x000	Reserved	0xE008 1000	0x5008 1000	0x0 1000	4 Kbyte	0x0000 1002
0x004	Timestamp generator	0xE008 2000	0x5008 2000	0x0 2000	4 Kbyte	0x0000 2003
0x008	SWO	0xE008 3000	0x5008 3000	0x0 3000	4 Kbyte	0x0000 3003
0x00C	Trace Subsystem ROM table	0xE009 0000	0x5009 0000	0x1 0000	4 Kbyte	0x0001 0003
0x010	STM	0xE00A 0000	0x500A 0000	0x1 0000	4 Kbyte	0x0002 0003
0x014	C-A7_1 DBG	0xE00D 0000	0x500D 0000	0x5 0000	4 Kbyte	0x0005 0003
0x018	C-A7_1 PMU	0xE00D 1000	0x500D 1000	0x5 1000	4 Kbyte	0x0005 1003
0x01C	C-A7_2 DBG	0xE00D 2000	0x500D 2000	0x5 2000	4 Kbyte	0x0005 2003
0x020	C-A7_2 PMU	0xE00D 3000	0x500D 3000	0x5 3000	4 Kbyte	0x0005 3003
0x024	C-A7_1 CTI	0xE00D 8000	0x500D 8000	0x5 8000	4 Kbyte	0x0005 8003
0x028	C-A7_2 CTI	0xE00D 9000	0x500D 9000	0x5 9000	4 Kbyte	0x0005 9003
0x02C	C-A7_1 ETM	0xE00D C000	0x500D C000	0x5 C000	4 Kbyte	0x0005 C003
0x030	C-A7_2 ETM	0xE00D D000	0x500D D000	0x5 D000	4 Kbyte	0x0005 D003
0x034	Top of table	-	-	-	-	0x0000 0000
0x038 to 0xFC8	Reserved	-	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	-	See Table 570

The second table occupies a 4-Kbyte, 32-bit wide chunk of APB address space, from 0xE009 0000 to 0xE009 0FFC when accessed by the debugger, and 0x5009 0000 to 0x5009 0FFC when accessed from the system bus. It points to the CoreSight™ components located in the trace subsystem.

Table 569. Trace subsystem ROM table

Address offset in ROM table	Component name	Component base address (debugger)	Component base address (system bus)	Component address offset	Size	Entry
0x000	Trace funnel	0xE009 1000	0x5009 1000	0x1000	4 Kbyte	0x0000 1003
0x004	ETF	0xE009 2000	0x5009 2000	0x2000	4 Kbyte	0x0000 2003
0x008	TPIU	0xE009 3000	0x5009 3000	0x3000	4 Kbyte	0x0000 3003
0x00C	Trace CTI	0xE009 4000	0x5009 4000	0x4000	4 Kbyte	0x0000 4003
0x010	Top of table	-	-	-	-	0x0000 0000
0x018 to 0xFC8	Reserved	-	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	-	See *Table 570*

The top of each ROM table contains a number of read only registers, including the standard CoreSight™ component and peripheral identity registers, refer to *Table 570*.

Each debug component occupies one or more 4-Kbyte blocks of address space. This block of address space is referred to as the Debug Register File for the component.

The component address offset field of a ROM Table entry points to the start of the last 4-Kbyte block of the address space of the component. This block always contains the Component and Peripheral ID Registers for the component, starting at offset 0xFD0 from the start of the block. The 4-Kbyte count field, bits PIDR4[7:4], specifies the number of 4-Kbyte blocks for the component. Therefore, the process for finding the start of the address space for a component is:

1. Read the ROM Table entry for the component, and extract the Address offset for the component. The Address offset is bits [31:12] of the ROM Table entry.
2. Use the Address offset, together with the base address of the ROM Table, to calculate the base address of the component. The component base address is:

   Component_Base_Address = ROM_Base_Address + (Address_Offset << 12)

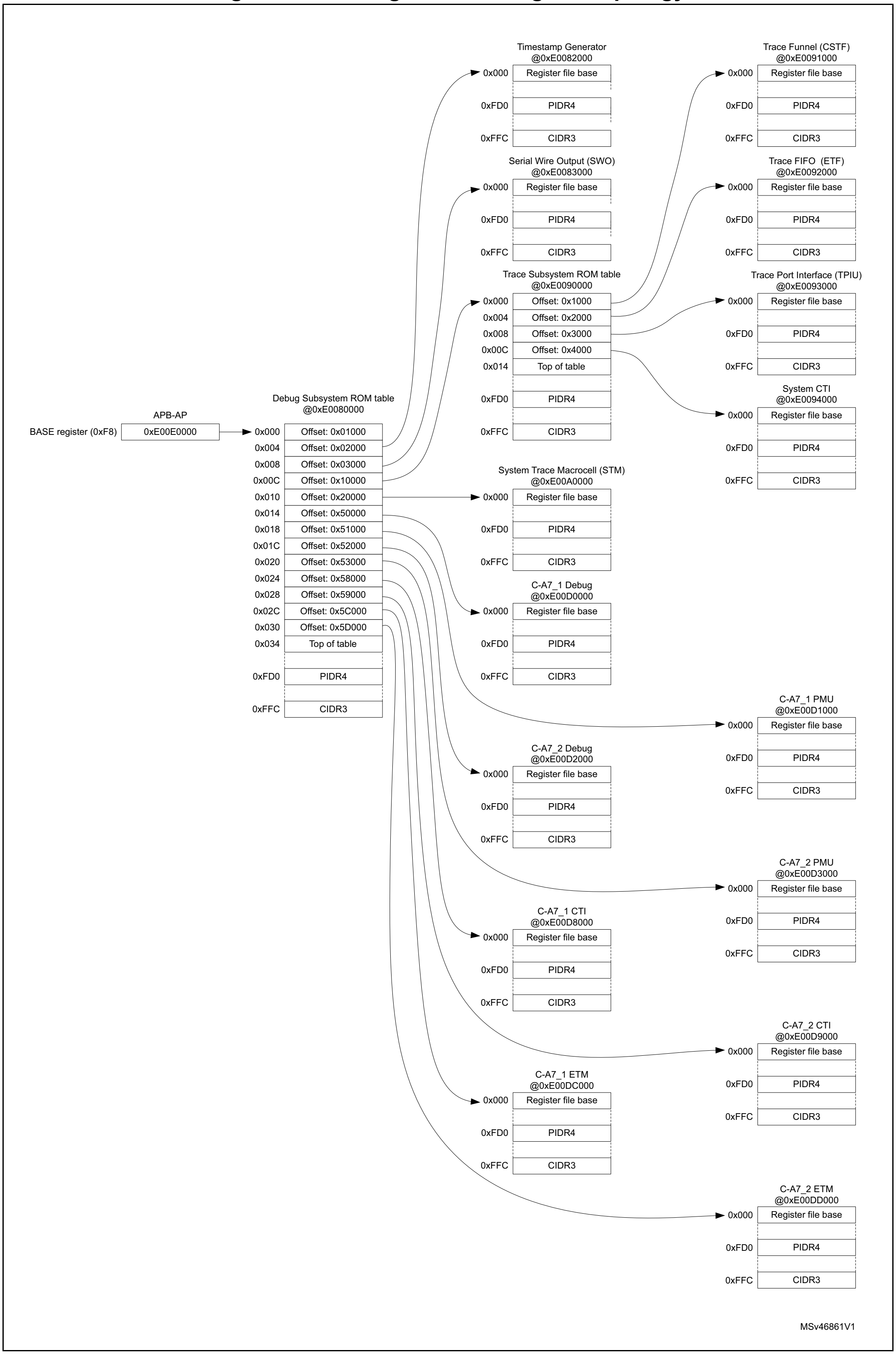
   Component_Base_Address is the start address of the final 4-Kbyte block of the address space for the component.
3. Read the Peripheral ID4 Register for the component. The address of this register is:

   Peripheral_ID4_address = Component_Base_Address + 0xFD0
4. Extract the 4-Kbyte count field, bits [7:4], from the value of the Peripheral ID4 Register.
5. Use the 4-Kbyte count value to calculate the start address of the address space for the component. If the 4-Kbyte count field is 0000, indicating a count value of 1, the address space for the component starts at the Component_Base_Address obtained at stage 2.

The topology for the CoreSight™ components on the debug APB is shown in *Figure 788*.

For more information on the use of the ROM table, refer to the *Arm® Debug Interface Architecture Specification [1]*.

Figure 788. Debug APB CoreSight™ topology



ROM table registers

SYSROM memory type register (SYSROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x0: no system memory is present on this bus

SYSROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

SYSROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x00: ROM table part number

SYSROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x5: ROM table part number

SYSROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

SYSROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

SYSROM CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

SYSROM CoreSight peripheral identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

SYSROM CoreSight component identity register 0 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

SYSROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

ROM table registers

Table 570. ROM table register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SYSROM_ MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4	SYSROM_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	SYSROM_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	SYSROM_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	SYSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									0	0	0	0	0	1	0	1
0xFE8	SYSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	0
0xFEC	SYSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SYSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SYSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0

Table 570. ROM table register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	SYSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SYSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.2 CoreSight™ global timestamp generator (TSGEN)

The CoreSight™ Global Timestamp Generator contains a 64-bit counter that provides a common timing reference for all of the trace sources in the system, namely the ETM, ITM and STM. These components insert timestamps in the trace streams that allow the trace analyser to recover the chronological order of trace packets, which can be lost when multiple trace sources are multiplexed into one stream at the funnels.

The TSG registers are accessible over the Debug APB. This allows the debugger or debug software to:

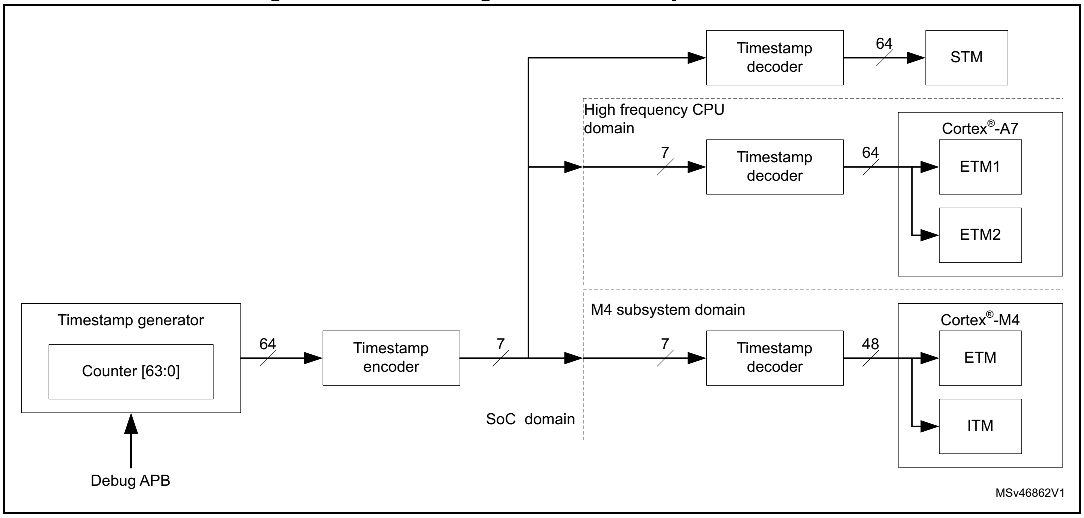
- Start and stop the timestamp incrementing.
- Cause the timestamp counter to stop when debug state is entered. To enable this feature, the trace subsystem CTI must be configured to signal to the timestamp generator when system-wide debug state has been entered (meaning that all CPU cores are halted).
- Read the current timestamp value.
- Change the current timestamp value. The timestamp counter must be halted while it is changed. When the timestamp value is changed, the timestamp generator resynchronizes all the trace sources.
- Change the reported timestamp increment.

For more information on the CoreSight™ Timestamp Generator CoreSight™ component, refer to the *Arm® CoreSight™ SoC-400 Technical Reference Manual [2]*.

The timestamp generator is located in the ck_apb_dbg domain, and the timestamp is distributed to the Cortex®-A7 and Cortex®-M4. To simplify the distribution over clock domain boundaries, the 64-bit timestamp is encoded in 7 bits, then decoded in the destination power domain. The timestamp distribution is shown in *Figure 789*.

Whenever the frequency of the TSGEN, Cortex®-A7 or Cortex®-M4 changes, the ETM of the respective processor receives a pulse on the TSCLKCHANGE input.

Figure 789. CoreSight™ timestamp distribution



Timestamp generator registers

TSG counter control register (TSG_CNTCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDBG	EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HDBG**: halt on debug
0: normal operation
1: halts counter when system-wide debug state is detected (via CTI)

Bit 0 **EN**: enable
0: counter disabled
1: counter enabled and incrementing

TSG counter status register (TSG_CNTSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBGH**: debug halted
0: normal operation
1: counter halted due to system-wide debug state

Bit 0 Reserved, must be kept at reset value.

TSG current counter value lower register (TSG_CNTCVL)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_L_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU_L_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU_L_32[31:0]**: timestamp counter current value (least significant 32 bits)

To change the current timestamp value, write the least significant 32 bits of the new value to this register before writing the most significant 32 bits to TSG_CNTCVU. The timestamp value is not changed until the TSG_CNTCVU register is written to.

Note: Bit EN of register TSG_CNTCR must be cleared before writing to register TSG_CNTCVL.

TSG current counter value upper register (TSG_CNTCVU)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_U_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU_U_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU_U_32[31:0]**: timestamp counter current value (most significant 32 bits)

To change the current timestamp value, write the least significant 32 bits of the new value to TSG_CNTCVL before writing the most significant 32 bits to this register. The 64-bit timestamp value is updated with the value from both writes when this register is written to.

Note: Bit EN of register TSG_CNTCR must be cleared before writing to register TSG_CNTCVU.

TSG base frequency ID register (TSG_CNTFID0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]**: increment frequency of TSG counter in Hz

This field must be programmed with the trace generator clock frequency whenever it changes.

TSG CoreSight peripheral identity register 4 (TSG_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TSG CoreSight peripheral identity register 0 (TSG_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x01: TSG part number

TSG CoreSight peripheral identity register 1 (TSG_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x1: TSG part number

TSG CoreSight peripheral identity register 2 (TSG_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

TSG CoreSight peripheral identity register 3 (TSG_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

TSG CoreSight component identity register 0 (TSG_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

TSG CoreSight peripheral identity register 1 (TSG_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0xF: CoreSight™ Soc-400 component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: common ID value

TSG CoreSight component identity register 2 (TSG_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

TSG CoreSight component identity register 3 (TSG_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Timestamp generator registers map

Table 571. TSGEN register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TSG_CNTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDBG	EN
	Reset value																															0	0
0x004	TSG_CNTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
	Reset value																															0	
0x008	TSG_CNTCVL	CNTCVU_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TSG_CNTCVU	CNTCVU_U_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TSG_CNTFID0	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFD0	TSG_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	TSG_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Table 571. TSGEN register map and reset values (continued)

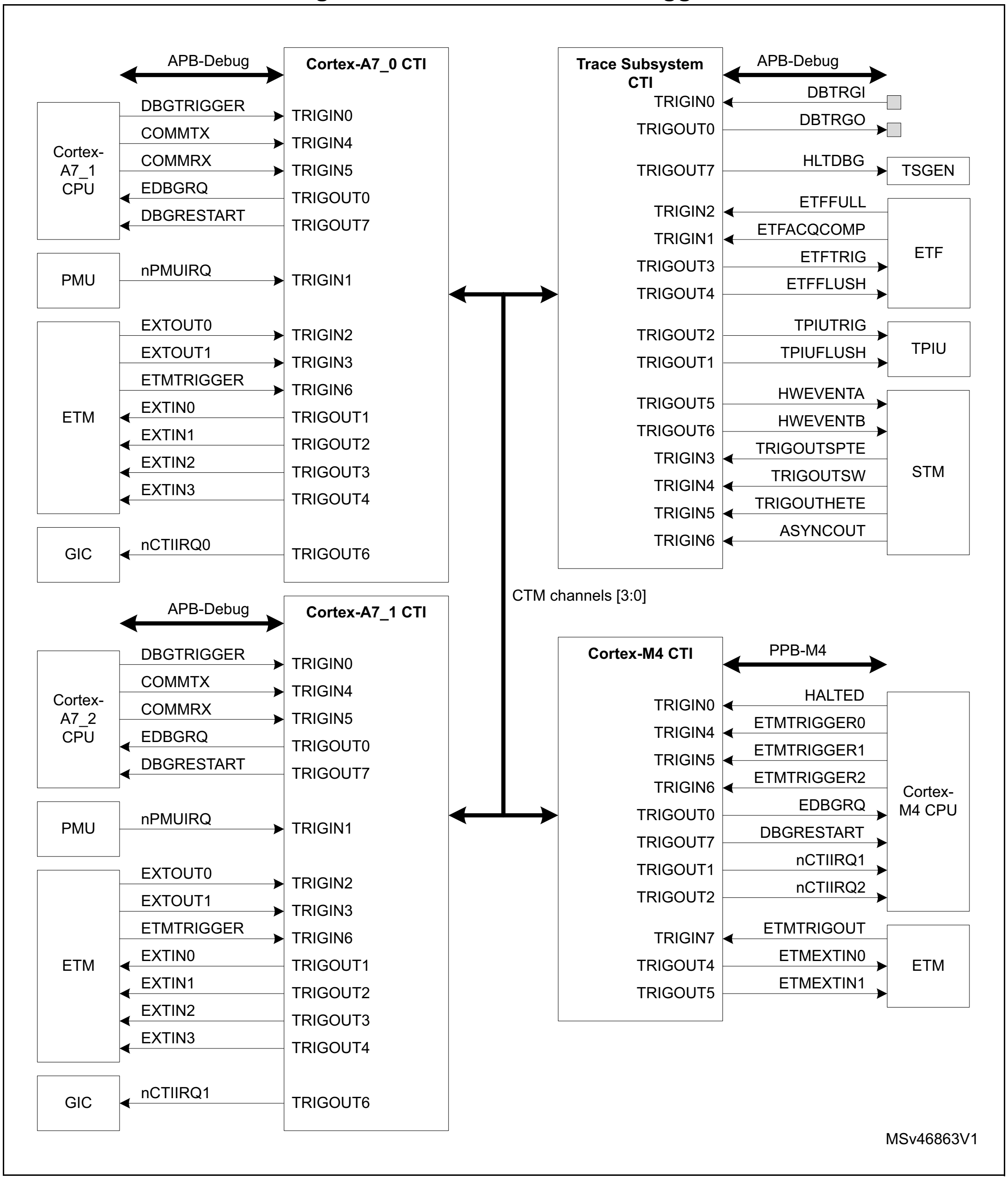
Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD8	TSG_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	TSG_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	TSG_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	TSG_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	1
0xFE8	TSG_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	TSG_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TSG_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TSG_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	TSG_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TSG_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.3 Cross trigger interface (CTI) and matrix (CTM)

The cross trigger interface (CTI) and cross trigger matrix (CTM) together form the CoreSight™ embedded cross trigger feature. There are four CTI components, one in the trace subsystem, one for each Cortex®-A7, and one dedicated to the Cortex®-M4. The CTIs are connected to each other via the CTM. The trace subsystem CTI and the Cortex®-A7

CTIs are accessible to the debugger via the debug APB. The Cortex®-M4 CTI is accessible on its private peripheral bus (PPB) via the Cortex®-M4 AHB access port.

Figure 790. Embedded cross trigger



The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a breakpoint reached in one of the processor cores can stop the other processor, or a transition detected on an external trigger input can start code trace.

Each CTI has up to 8 trigger inputs and 8 trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in *Table 572* to *Table 577*.

Table 572. Trace Subsystem CTI inputs

	Source signal	Source component	Comments
0	DBTRIGI	GPIO	External trigger input - allows an external signal to generate a debug event
1	ETFACQCOMP	ETF	ETF capture finished - allows a debug event to be generated when the trace FIFO is empty
2	ETFFULL	ETF	ETF full flag - allows a debug event to be generated when the trace FIFO is full
3	TRIGOUTSPTE	STM	The STM asserts this signal for one clock cycle when a trigger event is detected on a match using the STMSPTER.
4	TRIGOUTSW	STM	The STM asserts this signal for one clock cycle when a trigger event is generated on writes to a TRIG location in the extended stimulus port registers.
5	TRIGOUTHETE	STM	The STM asserts this signal for one clock cycle when a trigger event is detected on a match using the STMHETER.
6	ASYNCOUT	STM	Alignment synchronization signal. The STM asserts this signal for one clock cycle when an ASYNC-VERSION-FREQ sequence is output on the ATB interface, and the ASYNCOUT signal can be used for cross-triggering.
7	-	-	Not used

Table 573. Trace Subsystem CTI outputs

	Output signal	Destination component	Comments
0	DBTRIGO	GPIO	External I/O trigger output - allows monitoring of events on the external DBTRIGO pin
1	TPIUFLUSH	TPIU	Trace port flush trigger - causes the TPIU FIFO to be flushed
2	TPIUTRIG	TPIU	Trace Port enable trigger - starts trace output on the external trace port
3	ETFTRIG	ETF	ETF enable trigger - starts filling the Trace FIFO
4	ETFFLUSH	ETF	ETF flush trigger - causes the Trace FIFO to be flushed
5	HWEVENTA	STM	STM hardware event
6	HWEVENTB	STM	STM hardware event
7	HLTDBG	TSGEN	Halts timestamp in debug - freezes the TSGEN timestamp counter

Table 574. Cortex®-A7_1/2 CTI inputs

	Source signal	Source component	Comments
0	DBGTRIGGER	Cortex®-A7 CPU	Pulsed high when the core enters in debug state
1	nPMUIRQ	PMU	Performance monitor interrupt
2	EXTOUT0	ETM	ETM external trigger output
3	EXTOUT1	ETM	ETM external trigger output
4	COMMTX	Cortex®-A7 CPU	Debug comm port transmit
5	COMMRX	Cortex®-A7 CPU	Debug comm port receive
6	ETMTRIGGER	ETM	ETM trigger
7	-	-	Not used

Table 575. Cortex®-A7_1/2 CTI outputs

	Output signal	Destination component	Comments
0	EDBGRQ	Cortex®-A7 CPU	CPU halt request - puts CPU in debug mode
1	EXTIN0	ETM	ETM external input
2	EXTIN1	ETM	ETM external input
3	EXTIN2	ETM	ETM external input
4	EXTIN3	ETM	ETM external input
5	-	-	Not used
6	nCTIIRQ	GIC	CTI interrupt
7	DBGRESTART	Cortex®-A7 CPU	CPU restart request - CPU exits debug mode

Table 576. Cortex®-M4 CTI inputs

	Source signal	Source component	Comments
0	HALTED	M4 CPU	CPU halted - indicates CPU is in debug mode
1	-	-	Not used
2	-	-	Not used
3	-	-	Not used
4	ETMTRIGGER0	M4 DWT	Trace trigger - enables CPU execution trace
5	ETMTRIGGER1	M4 DWT	Trace trigger - enables CPU execution trace
6	ETMTRIGGER2	M4 DWT	Trace trigger - enables CPU execution trace
7	ETMTRIGOUT	M4 ETM	ETM triggered - indicates trace active

Table 577. Cortex®-M4 CTI outputs

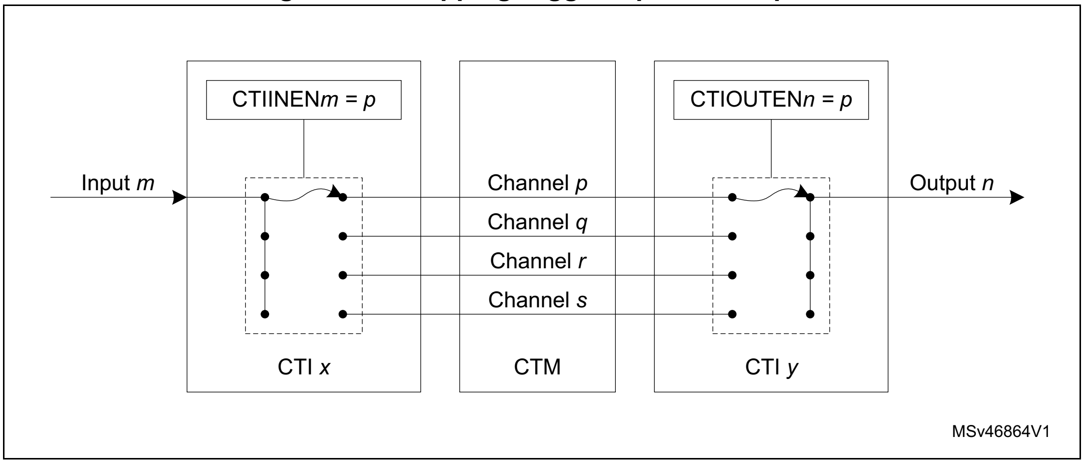
	Source signal	Destination component	Comments
0	EDBGRQ	M4 CPU	CPU halt request -puts CPU in debug mode
1	nIRQ1	M4 NVIC	CTI interrupt
2	nIRQ2	M4 NVIC	CTI interrupt
3	-	-	Not used
4	ETMEXTIN0	M4 ETM	ETM external input
5	ETMEXTIN1	M4 ETM	ETM external input
6	-	-	Not used
7	DBGRESTART	M4 ETM	CPU restart request - CPU exits debug mode

There are four event channels in the cross trigger matrix, which allows up to four, parallel, bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number *m* on CTI *x* to output number *n* on CTI *y*, the input must be connected to an event channel *p* using the CTI_INEN*m* register of CTI *x*. The same channel *p* must be connected to the output using the CTI_OUTEN*n* register of CTI *y*.

Note: The above connection settings are also valid if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four). As a result, an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

Figure 791. Mapping trigger inputs to outputs



Example configuration:

When either the MCU or the MPU hits a breakpoint, stop the other processor. Restart the two processors synchronously.

To stop both cores when either core stops requires the debug status output of each core (that is HALTED for the Cortex®-M4, DBGTRIGGER for the Cortex®-A7) to be connected to the EDBGRQ input of the opposite core.

Referring to *Table 574* and *Table 576*, we see that the DBGTRIGGER signal from the Cortex®-A7_1 core is connected to input 0 of the Cortex®-A7_1 CTI, and the HALTED signal from the Cortex®-M4 core is connected to the same input on the Cortex®-M4

CTI. Hence we program the CTI_IEN0 register on each CTI to connect these inputs to a CTM channel (eg. channel 0).

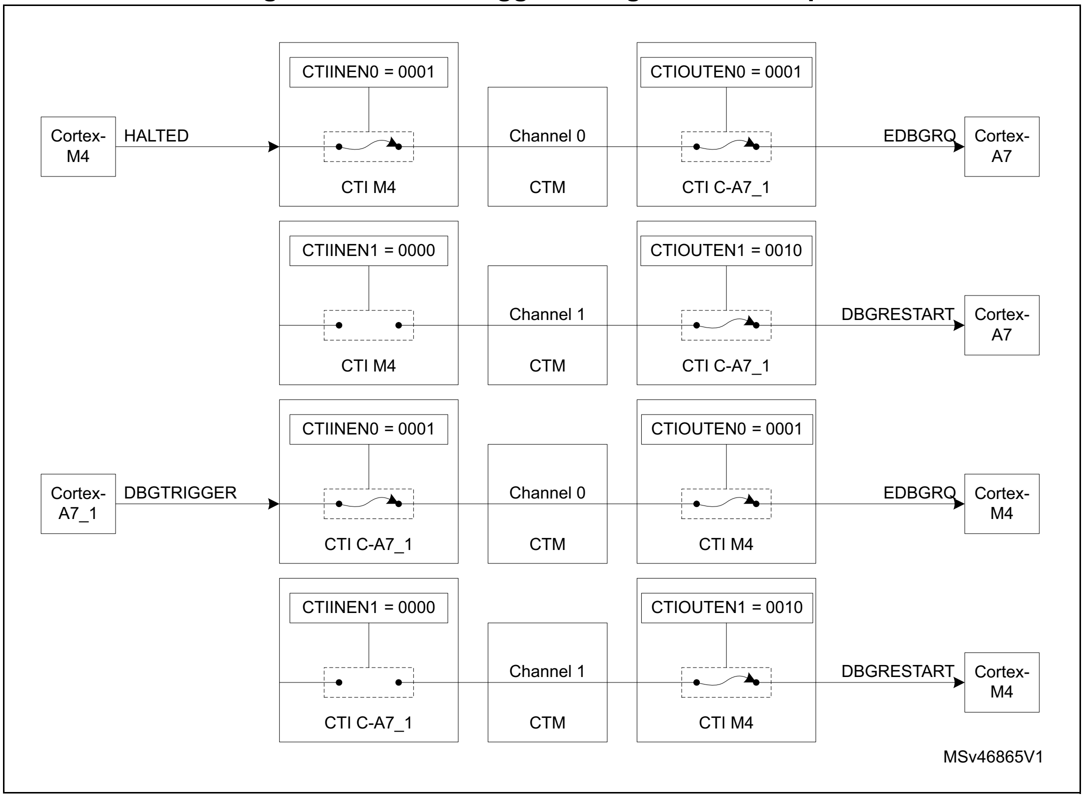
From *Table 575* and *Table 577* we see that the EDBGRQ signals to the CPUs are connected to output 0 of the respective CTIs. So we program the CTI_OUTEN0 register on each CTI to connect these outputs to the same CTM channel.

To restart both cores simultaneously the debugger must use the CTI_APPPULSE register in either of the CTIs. This allows the debugger to generate a pulse on any of the four ETM channels. The channel must be connected to the DBGRESTART signal of both cores.

From *Table 575* and *Table 577* we see that the DBGRESTART signals to the CPUs are connected to output 1 of the respective CTIs. So we program the CTI_OUTEN1 register on each CTI to connect these outputs to an unused CTM channel (eg. channel 1).

The above configuration is illustrated in *Figure 792*.

Figure 792. Cross trigger configuration example



To force the processors to restart simultaneously, the debugger should use the following procedure:

1. Clear the debug request by writing 0x01, then 0x00, to the CTI_INTACK register in each CTI.
2. Cause a pulse on channel 1 by writing 0x02 to the CTI_APPPULSE register in either CTI. This will generate a restart request to both processors.

   Note that the debugger can also force both cores to stop simultaneously by writing 0x01 to the CTI_APPPULSE register in either CTI, which generates a pulse on channel 0.

For more information on the cross trigger interface CoreSight™ component, refer to the *Arm® CoreSight™ SoC-400 Technical Reference Manual [2]*.

Cross trigger interface registers

The register file base address for each CTI is defined by the ROM table for the bus to which it is connected. The registers are the same for each CTI.

CTI control register (CTI_CONTROL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN:** global enable
0: cross-triggering disabled
1: cross-triggering enabled

CTI trigger acknowledge register (CTI_INTACK)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: trigger acknowledge
There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: channel event set
Read:
XXX0: channel 0 event inactive
XXX1: channel 0 event active
XX0X: channel 1 event inactive
XX1X: channel 1 event active
X0XX: channel 2 event inactive
X1XX: channel 2 event active
0XXX: channel 3 event inactive
1XXX: channel 3 event active
Write:
XXX0: no effect
XXX1: sets event on channel 0
XX0X: no effect
XX1X: sets event on channel 1
X0XX: no effect
X1XX: sets event on channel 2
0XXX: no effect
1XXX: sets event on channel 3

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: channel event clear
XXX0: no effect
XXX1: clears event on channel 0
XX0X: no effect
XX1X: clears event on channel 1
X0XX: no effect
X1XX: clears event on channel 2
0XXX: no effect
1XXX: clears event on channel 3

CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPULSE[3:0]**: pulse channel event
This register clears itself immediately.
XXX0: no effect
XXX1: generates pulse on channel 0
XX0X: no effect
XX1X: generates pulse on channel 1
X0XX: no effect
X1XX: generates pulse on channel 2
0XXX: no effect
1XXX: generates pulse on channel 3

CTI trigger in enable x registers (CTI_INENx)

Address offset: 0x020+ 0x04 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: cross-trigger event enable / disable
Enables or disables a cross-trigger event on each of the four channels when CTITRIGINn is activated (n = 0 to 7).
XXX0: trigger n does not generate events on channel 0
XXX1: trigger n generates events on channel 0
XX0X: trigger n does not generate events on channel 1
XX1X: trigger n generates events on channel 1
X0XX: trigger n does not generate events on channel 2
X1XX: trigger n generates events on channel 2
0XXX: trigger n does not generate events on channel 3
1XXX: trigger n generates events on channel 3

CTI Trigger out enable x registers (CTI_OUTENx)

Address offset: 0x0A0 + 0x04 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: output trigger enable / disable
For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTn (n = 0 to 7).
XXX0: channel 0 events does not generate triggers on trigger output n
XXX1: channel 0 events generates triggers on trigger output n
XX0X: channel 1 events does not generate triggers on trigger output n
XX1X: channel 1 events generates triggers on trigger output n
X0XX: channel 2 events does not generate triggers on trigger output n
X1XX: channel 2 events generates triggers on trigger output n
0XXX: channel 3 events does not generate triggers on trigger output n
1XXX: channel 3 events generates triggers on trigger output n

CTI trigger in status register (CTI_TRGISTS)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: trigger input status
There is one bit of the register for each CTITRIGIN input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI Trigger out status register (CTI_TRGOSTS)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: trigger output status
There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel in status register (CTI_CHINSTS)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: channel input status
There is one bit of the register for each channel input. When a bit is set to 1, it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel out status register (CTI_CHOUTSTS)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: channel output status

There is one bit of the register for each channel output. When a bit is set to 1, it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATE)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

XXX0: channel 0 events do not propagate
XXX1: channel 0 events propagate
XX0X: channel 1 events do not propagate
XX1X: channel 1 events propagate
X0XX: channel 2 events do not propagate
X1XX: channel 2 events propagate
0XXX: channel 3 events do not propagate
1XXX: channel 3 events propagate

CTI claim tag set register (CTI_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: sets claim tag bits

Write:
- 0000: no effect
- xxx1: sets bit 0
- xx1x: sets bit 1
- x1xx: sets bit 2
- 1xxx: sets bit 3

Read:
- 0xF: indicates there are four bits in claim tag

CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: resets claim tag bits

Write:
- 0000: no effect
- xxx1: clears bit 0
- xx1x: clears bit 1
- x1xx: clears bit 2
- 1xxx: clears bit 3

Read:
- Returns current value of claim tag

CTI lock access register (CTI_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CTI register write access enable

Enables write access to some CTI registers by processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: enable write access

Other values: disable write access

CTI lock status register (CTI_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: CTI_LAR register size

Indicates the size of the CTI_LAR register.

0: 32-bit

Bit 1 **LOCKGRANT**: current lock status

This bit always reads as zero by an external debugger.

0: write access is permitted

1: write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST**: lock control existence

Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.

0: no lock control mechanism exists

1: lock control mechanism is implemented

CTI authentication status register (CTI_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x0: not implemented

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x2: disabled
0x3: enabled

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x2: disabled
0x3: enabled

CTI device configuration register (CTI_DEVID)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: number of ECT channels available
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]**: number of ECT triggers available
0x8: 8 trigger inputs and 8 trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: number of trigger input/output multiplexers
0x0: none

CTI device type Identifier register (CTI_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: indicates that this component is a cross-triggering component

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x4: indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x06: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x9: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x4: r0p5

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

CTI CoreSight component identity register 0 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Cross trigger interface registers map

Table 578. CTI register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x010	CTI_INTACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	CTI_APPSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0

Table 578. CTI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APULSE[3:0]			
	Reset value																													0	0	0	0
0x020	CTI_INEN0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x024	CTI_INEN1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x028	CTI_INEN2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x02C	CTI_INEN3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x030	CTI_INEN4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x034	CTI_INEN5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0

Table 578. CTI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	CTI_INEN6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x03C	CTI_INEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
	Reset value																													0	0	0	0
0x0A0	CTI_OUTEN0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0A4	CTI_OUTEN1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0A8	CTI_OUTEN2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0AC	CTI_OUTEN3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0B0	CTI_OUTEN4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0

Table 578. CTI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B4	CTI_OUTEN5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0B8	CTI_OUTEN6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x0BC	CTI_OUTEN7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
	Reset value																													0	0	0	0
0x130	CTI_TRIGISTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRIGOSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHISTATUS[3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOSTATUS[3:0]			
	Reset value																													0	0	0	0
0x140	CTI_GATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN [3:0]			
	Reset value																													1	1	1	1
0xFA0	CTI_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1

Table 578. CTI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA4	**CTI_CLAIMCLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB0	**CTI_LAR**	ACCESS_W[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFB4	**CTI_LSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	**CTI_AUTHSTAT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	1	0	1
0xFC8	**CTI_DEVID**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH [3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXMUXNUM[4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	**CTI_DEVTYPE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	**CTI_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	**CTI_PIDR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	**CTI_PIDR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	**CTI_PIDR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	**CTI_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	1	1	0

Table 578. CTI register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	1	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.4 Trace funnel (CSTF)

The trace funnel is a CoreSight™ component that combines the ATB buses from the five trace sources into one single ATB. The CSTF has five ATB slave ports, and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:

- S0: STM
- S1: Cortex®-A7_1 ETM
- S2: Cortex®-A7_2 ETM
- S3: Cortex®-M4 ETM
- S4: Cortex®-M4 ITM

The CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when trace is disabled. The arbitration works as follows:

- The arbiter selects the slave port with the highest assigned priority that has data valid
- Up to *min_hold_time* transfers are passed from the selected slave to the master port, where *min_hold_time* is programmable in the CONTROL register.
- A new arbitration is then performed

High priority should be assigned to slave ports connected to sources with a small amount of buffering, or where data loss can not be tolerated. Low priority should be assigned to less critical sources or those with large buffers.

For more information on the ATB Funnel CoreSight™ component, refer to the *Arm® CoreSight™ SoC-400 Technical Reference Manual [2]*.

Trace funnel registers

CSTF control register (CSTF_CTRL)

Address offset: 0x000

Reset value: 0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	ENS4	ENS3	ENS2	ENS1	ENS0
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **MIN_HOLD_TIME[3:0]**: number of transactions between each arbitration
0x0: 1 transaction
:
0xE: 15 transactions
0xF: reserved

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ENS4**: slave port S4 enable
0: disables port
1: enables port

Bit 3 **ENS3**: slave port S3 enable
0: disables port
1: enables port

Bit 2 **ENS2**: slave port S2 enable
0: disables port
1: enables port

Bit 1 **ENS1**: slave port S1 enable
0: disables port
1: enables port

Bit 0 **ENS0**: slave port S0 enable
0: disables port
1: enables port

CSTF priority register (CSTF_PRIORITY)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PRIPORT4[2:0]			PRIPORT3[2:0]			PRIPORT2[2:0]			PRIPORT1[2:0]			PRIPORT0[2:0]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **PRIPORT4[2:0]**: slave port S4 priority
0: highest priority
:
7: lowest priority

Bits 11:9 **PRIPORT3[2:0]**: slave port S3 priority
0: highest priority
:
7: lowest priority

Bits 8:6 **PRIPORT2[2:0]**: slave port S2 priority
0: highest priority
:
7: lowest priority

Bits 5:3 **PRIPORT1[2:0]**: slave port S1 priority
0: highest priority
:
7: lowest priority

Bits 2:0 **PRIPORT0[2:0]**: slave port S0 priority
0: highest priority
:
7: lowest priority

CSTF claim tag set register (CSTF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: sets claim tag bits

Write:
- 0000: no effect
- xxx1: sets bit 0
- xx1x: sets bit 1
- x1xx: sets bit 2
- 1xxx: sets bit 3

Read:
- 0xF: indicates that there are four bits in claim tag

CSTF claim tag clear register (CSTF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: resets claim tag bits

Write:
- 0000: no effect
- xxx1: clears bit 0
- xx1x: clears bit 1
- x1xx: clears bit 2
- 1xxx: clears bit 3

Read:
- Returns current value of claim tag

CSTF lock access register (CSTF_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CTI register write access enable

Enables write access to some CTI registers by processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: enable write access

Other values: disable write access

CSTF lock status register (CSTF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE:** CSTF_LAR register size

Indicates the size of the CSTF_LAR register.

0: 32-bit

Bit 1 **LOCKGRANT:** current lock status

This bit always reads as zero by an external debugger.

0: write access is permitted

1: write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST:** lock control existence

Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.

0: no lock control mechanism exists

1: lock control mechanism is implemented

CSTF authentication status register (CSTF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x0: not implemented

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x0: not implemented

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

CSTF CoreSight device identity register (CSTF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SCHEME[3:0]**: priority scheme
0x2: static priority

Bits 3:0 **PORTCNT[3:0]**: number of input ports connected
0x4: four input ports

CSTF CoreSight device identity register (CSTF_TYPEID)

Address offset: 0xFCC

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: device type identifier
0x12: trace funnel

CSTF CoreSight peripheral identity register 0 (CSTF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x08: CSTF part number

CSTF CoreSight peripheral identity register 1 (CSTF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x9: CSTF part number

CSTF CoreSight peripheral identity register 2 (CSTF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x1: r0p1

Bit 3 **JEDEC:** JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CSTF CoreSight peripheral identity register 3 (CSTF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CSTF CoreSight peripheral identity register 4 (CSTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CSTF CoreSight component identity register 0 (CSTF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

CSTF CoreSight peripheral identity register 1 (CSTF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

CSTF CoreSight component identity register 2 (CSTF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

CSTF CoreSight component identity register 3 (CSTF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Trace funnel registers map

Table 579. CSTF register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**CSTF_CTRL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	ENS3	ENS2	ENS1	ENS0
	Reset value																					0	0	1	1					0	0	0	0
0x004	**CSTF_PRIORITY**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT3[2:0]			PRIPORT2[2:0]			PRIPORT1[2:0]			PRIPORT0[2:0]		
	Reset value																					0	1	1	0	1	0	0	0	1	0	0	0

Table 579. CSTF register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA0	CSTF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	CSTF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB0	CSTF_LAR	ACCESS_W[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFB4	CSTF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	CSTF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	CSTF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
	Reset value																									0	0	1	0	0	1	0	0
0xFCC	CSTF_TYPEID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value																									0	0	0	1	0	0	1	0
0xFD0	CSTF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	CSTF_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	CSTF_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	CSTF_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Table 579. CSTF register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	**CSTF_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	0	0	0
0xFE4	**CSTF_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	**CSTF_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	**CSTF_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**CSTF_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**CSTF_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**CSTF_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**CSTF_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.5 Embedded trace FIFO (ETF)

The ETF is an 8-kbyte memory which captures trace data from five trace sources, namely the ETM of each CPU core, the ITM of the Cortex®-M4, and the STM. The ETF is a configuration of the CoreSight™ Trace Memory Controller component.

The ETF can be used in three modes (selected in the Mode register):

1. Hardware FIFO mode

   The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured into the Trace RAM and when full, the incoming trace stream is stalled. When the Trace buffer is not empty, trace data is drained out through the ATB master interface to the TPIU.

   In this mode, the role of the FIFO is to smooth the flow of trace information arriving at the trace port. Since the trace data can be very bursty in nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, which can then be sized according to the average rate rather than the peak. The trace is stored off-chip in real time by the trace port analyser tool, and so the trace log can be very big.

2. Software FIFO mode

   The trace memory is used as a FIFO that can be read through the RRD Register while trace is being captured. Trace data is captured into the Trace RAM and when full, the incoming trace stream is stalled.

   This mode allows the trace to be transferred by DMA into the system memory, or to a high speed interface (SPI, USB etc), or even monitored by software running on one of the cores. Note that unlike the hardware FIFO mode, this mode is invasive, since it uses system resources which are shared by the processors.

3. Circular buffer mode

   The trace memory is used as a circular buffer. Trace data is captured into the Trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition occurs.

   In this mode, the ETF stores the trace data on-chip, so the trace log size is limited to that of the ETF SRAM, 8 kbytes in this case. Being a circular buffer, if the FIFO becomes full, incoming trace data will overwrite the oldest stored data, which will be lost. Therefore the contents of the trace buffer represent the most recent activity of the processor(s), up to the point when the buffer was stopped, rather than all the activity since the trace was started.

   There are three possible methods to read out the buffer contents once the trace has stopped:

   – via the Trace Port - with the TPIU enabled, the contents of the buffer are output over the trace port. This can be done by setting the DrainBuffer bit in the ETF_FFCR register.
   – via the Debug Port - the debugger can read the buffer via the RRD register which is accessible over the debug APB.
   – by software - either processor can read the buffer via the RRD register, since the debug APB is accessible from the system bus. This could also be performed by software triggered DMA transfers. No hardware DMA request is available so the FULL flag can be used to trigger an interrupt to the appropriate core via the cross trigger. The software can then read the ETF_CBUFLVL register to find out the number of data to be read.

For more information on the Trace Memory Controller CoreSight™ component, refer to the *Arm® CoreSight™ Trace Memory Controller Technical Reference Manual [3]*.

Embedded trace FIFO registers

ETF RAM size register (ETF_RSZ)

Address offset: 0x004

Reset value: 0x0000 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RSZ[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSZ[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **RSZ[30:0]**: size of the RAM in 32-bit words

0x800: 8 Kbytes

ETF status register (ETF_STS)

Address offset: 0x00C

Reset value: 0x0000 001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EMPTY:** trace FIFO empty

This bit is valid only when the TCEN bit of the ETF_CTL register is high. This bit reads as zero when TCEN is low.

0: trace FIFO contains data

1: trace FIFO is empty

Note: EMPTY does not inform about ETF pipeline emptiness which is indicated by FTEMPTY.

Bit 3 **FTEMPTY**: formatter empty

This bit is set when trace capture has stopped, and all internal pipelines and buffers have drained. Unlike READY, it is not affected by buffer drains. The ACQCOMP output reflects the value of this bit.

Bit 2 **READY**: ETF ready

This bit is set when trace capture has stopped and all internal pipelines and buffers have drained (STOPPED or DISABLED mode)

Bit 1 **TRIGD**: triggered

Bit triggered is set when trace capture is in progress and the TMC has detected a trigger event. This bit is cleared when leaving disabled state.

This bit is operational only in the circular buffer mode. In all other modes, this bit is always low.

This bit does not indicate that a trigger has been embedded in the formatted output trace data from the TMC. Trigger indication on the output trace stream is determined by the programming of the Formatter and Flush Control Register, ETF_FFCR.

Bit 0 **FULL**: trace buffer full

In circular buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until the TCEN bit of the ETF_CTL register is cleared and set.

In software and hardware FIFO modes, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the ETF_BUFWM register, that is, Fill level >= MEM_SIZE - BUFWM.

This bit is cleared when leaving the disabled state. The FULL output reflects the value of this register bit.

ETF RAM read data register (ETF_RRD)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RRD[31:0]**: RAM read data

Circular buffer mode: When in stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer. When all of the trace buffer has been read, the EMPTY bit in the ETF_STS Register is set, and subsequent reads return 0xFFF FFFF. Reading this register when not in stopped state returns 0xFFFF FFFF.

Software FIFO mode: Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, the data returned is 0xFFFF FFFF.

Hardware FIFO mode: Reading this register returns 0xFFFF FFFF.

ETF RAM read pointer register (ETF_RRP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	RRP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RRP[12:0]**: RAM read pointer

The RAM read pointer register contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via the ETF_RRD register. The pointer can be programmed with a byte address, 64-bit aligned (bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been written. When the pointer reaches its maximum value, it wraps around.

This register can only be written in DISABLED state. It can be read in DISABLED state, in STOPPED state in circular buffer mode and SW FIFO mode, and also in RUNNING and STOPPING states in SW FIFO mode.

ETF RAM write pointer register (ETF_RWP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	RWP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RWP[12:0]**: RAM write pointer

The RAM rite Pointer Register contains the value of the write pointer that is used to write entries into the trace memory over the APB interface via the ETF_RWD register. The pointer can be programmed with a byte address, 64-bit aligned (meaning that bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been read. When the pointer reaches its maximum value, it wraps around.

This register can only be written in DISABLED state. It can be read in DISABLED state, in STOPPED state in circular buffer mode and SW FIFO mode, and also in RUNNING and STOPPING states in SW FIFO mode.

ETF trigger counter register (ETF_TRG)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **TRG[10:0]**: trigger counter

In circular buffer mode, specifies the number of 32-bit words to capture in the trace RAM following the detection of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream, ATID =7'h7D. On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETF behavior is controlled by the ETF_FFCR register.

The number of 32-bit words written into the Trace RAM following the trigger is the value stored in this register, plus one. This register is ignored when the ETF is in software FIFO mode or hardware FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches zero. When the trigger counter has reached zero, it remains at zero until it is re-programmed with a write to this register.

This register is cleared when READY goes HIGH, so that the state of the counter when trace capture has stopped does not affect a subsequent trace capture session. Writing to this register when not in DISABLED state results in unpredictable behavior.

A read access to this register is permitted at any time when in DISABLED state, or in circular buffer mode. A read access returns the current value of the trigger counter.

ETF control register (ETF_CTL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TCEN:** trace capture enable

When writing:

0: disables trace capture (moves from RUNNING, STOPPING or STOPPED state into DISABLING or DISABLED state)

1: enables trace capture (moves from DISABLED state into RUNNING state)

When reading, this bit is low when in DISABLING or DISABLED states, and high otherwise.

ETF RAM write data register (ETF_RWD)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **RWD[31:0]**: RAM write data

When in DISABLED state, a write to this register stores data at the location pointed to by the RWP. Writes to this register when not in disabled state are ignored. When a full memory width (64-bit) of data has been written, the data is written to memory and the RAM write pointer is incremented to the next memory word.

This register is used for test purposes.

ETF mode register (ETF_MODE)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MODE[1:0]**: operation mode

0: circular buffer mode. In this mode, the trace memory is used as a circular buffer. Trace data is captured into the trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition has occurred.

1: software FIFO mode. In this mode, the trace memory is used as a FIFO that can be read through the ETF_RRD Register while trace is being captured. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled.

2: hardware FIFO mode. In this mode, the trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured into the trace RAM and when full, the incoming trace stream is stalled. When the trace buffer is non-empty, trace data is drained out through the ATB master interface to the TPIU.

3: reserved

ETF latched buffer fill level register (ETF_LBUFLVL)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LBUFLEVEL[11:0]**: latched buffer fill level

Reading this register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.

When entering disabled state, this register retains its last value. While in disabled state, reads from this register do not affect its value. When exiting disabled state, the ETF_LBUFLVL register is cleared.

This register is used for performance analysis of the trace system.

ETF current buffer fill level register (ETF_CBUFLVL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CBUFLEVEL[11:0]**: current buffer fill level

Reading this register returns the current fill level of the trace memory in 32-bit words.

This register is cleared when ETF_CTL.TCEN is low.

ETF buffer level watermark register (ETF_BUFWM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **BUFWM[10:0]**: buffer level watermark

The value programmed into this register indicates the required threshold vacancy level in 32-bit words in the trace memory. When the space in the FIFO is less than or equal to this value, that is, Fill level >= MEM_SIZE - BUFWM, the FULL output is pulled high and the FULL bit in the ETF_STS register is set.

This register is used only in software FIFO and hardware FIFO modes. In circular buffer mode, this functionality can be obtained by programming ETF_RWP to the required vacancy trigger level, so that when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.

The maximum value that can be written into this register is MEM_SIZE - 1; in this case, the FULL bit output is asserted after the first 32-bit word is written to trace memory.

Writing to this register other than when in DISABLED state results in unpredictable behavior.

ETF formatter and flush status register (ETF_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FTSTOPPED:** formatter stopped

This bit behaves in the same way as the FTEMPTY bit in the ETF_STS register.

Bit 0 **FLINPROG:** flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the ETF_FFCR register, or requested by the ATB master port.

0: no flush in progress

1: flush in progress

ETF formatter and flush control register (ETF_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRIGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	rw	rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DRAINBUF:** drain buffer

This bit is used to enable draining of the trace data through the ATB master interface after the formatter has stopped. This is useful in circular buffer mode to capture trace data into trace memory and then to drain the captured trace through the ATB master interface.

Writing a one to this bit when in stopped state starts the drain of the contents of the trace buffer through the ATB master interface. This bit always reads as zero. The READY bit in the ETF_STS register goes low while the drain is in progress.

This bit is functional only when the ETF is in circular buffer mode and formatting is enabled, that is if the ENFT bit in the ETF_FFCR register is set. Setting this bit when the ETF is in any other mode, or when not in stopped state, results in unpredictable behavior.

When trace capture is complete in circular buffer mode, all of the captured trace must be retrieved from the trace memory through the same mechanism, either read all trace data out through ETF_RRD reads, or drain all trace data by setting the DRAINBUF bit. Setting the DRAINBUF bit after some of the captured trace has been read out through ETF_RRD results in unpredictable behavior.

Bit 13 **STPONTRGEV:** stop on trigger event

0: no effect

1: stop trace capture when a trigger event occurs

Note: Enabling the ETF in software FIFO mode or hardware FIFO mode with this bit set results in unpredictable behavior.

Bit 12 **STOPONFL:** stop on flush

0: no effect

1: stops trace capture when flush is completed

If a flush is initiated by the ATB master interface, its completion does not lead to a formatter stop regardless of the value programmed in this bit.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGONFL:** trigger on flush

0: no effect

1: indicates a trigger in the trace stream when flush is completed

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

If a flush is initiated by the ATB master interface, its completion does not lead to a trigger indication regardless of the value programmed in this bit.

Bit 9 **TRGONTRGEV:** trigger on trigger event
0: no effect
1: indicates a trigger in the trace stream when trigger event occurs
If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.
This bit is not supported in software FIFO mode or hardware FIFO mode.

Bit 8 **TRGONTRGIN:** trigger on trigger in
0: no effect
1: indicates a trigger in the trace stream when a rising edge is detected on the TRIGIN input
If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FLUSHMAN:** manual flush
0: no effect
1: flushes the trace FIFO and pipeline
This bit is cleared automatically when the flush completes. If the TCEN bit in the ETF_CTL register is 0, writes to this bit are ignored.

Bit 5 **FONTRGEV:** flush on trigger event
0: no effect
1: flushes the trace FIFO and pipeline if a trigger event occurs
This bit is not supported in software FIFO mode or hardware FIFO mode. If STPONTRGEV is set, this bit is ignored.

Bit 4 **FONFLIN:** flush on flush in
0: no effect
1: flushes the trace FIFO and pipeline when a rising edge is detected on the FLUSHIN input

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENTI**: enable trigger insertion
Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 8'h00 with ATID 7'h7D in the trace stream. Trigger indication on the trace stream is additionally controlled by the register bits TRIGONFL, TRGONTRGEV, and TRGONTRGIN in the ETF_FFCR Register. This bit can only be changed when READY is high, and TCEN is low. This bit takes effect only when the ENFT register bit in this register is set. If the ENTI bit is set to high when ENFT is low, it results in formatting being enabled.

Bit 0 **ENFT**: enable formatting
0: formatting is disabled. Incoming trace data is assumed to be from a single trace source
1: formatting is enabled
If multiple ATIDs are received by the ETF when trace capture is enabled and the formatter is disabled, it results in interleaving of trace data. Disabling of formatting is supported only in circular buffer mode. If the ETF is enabled in a mode other than circular buffer mode with ENFT low, it results in formatting being enabled. If the ENTI bit is set to high when ENFT is low, it results in formatting being enabled.
This bit is ignored when in disabled state.

ETF periodic synchronisation counter register (ETF_PSCR)

Address offset: 0x308

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PSCOUNT[4:0]**: synchronization counter reload value

PSCOUNT determines the reload value of the synchronization counter. The reload value takes effect the next time the counter reaches zero. Reads from this register return the reload value programmed into this register. This register is set to 0xA on reset, corresponding to a synchronization period of 1024 bytes.

0x0: synchronization disabled

0x1-0x6: reserved

0x7-0x1B: synchronization period is $2^{PSCOUNT}$ bytes

Others: reserved

ETF claim tag set register (ETF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: sets claim tag bits

Write:
- 0000: no effect
- xxx1: sets bit 0
- xx1x: sets bit 1
- x1xx: sets bit 2
- 1xxx: sets bit 3

Read:
- 0xF: indicates there are four bits in claim tag

ETF claim tag clear register (ETF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: resets claim tag bits

Write:
- 0000: no effect
- xxx1: clears bit 0
- xx1x: clears bit 1
- x1xx: clears bit 2
- 1xxx: clears bit 3

Read:
- Returns current value of claim tag

ETF lock access register (ETF_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETF register write access enable

Enables write access to some ETF registers by processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: enable write access

Other values: disable write access

ETF lock status register (ETF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE:** ETF_LAR register size
Indicates the size of the ETF_LAR register.
0: 32-bit

Bit 1 **LOCKGRANT:** current lock status
This bit always reads as zero by an external debugger.
0: write access is permitted
1: write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST:** lock control existence
Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.
0: no lock control mechanism exists
1: lock control mechanism is implemented

ETF authentication status register (ETF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x0: not implemented

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x0: not implemented

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

ETF device configuration register (ETF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0380

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	MEMWIDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **MEMWIDTH[2:0]**: memory interface data bus width
0x3: 64 bits (corresponds to 32-bit ATB data)

Bits 7:6 **CONFIGTYP[1:0]**: configuration type of component (ETB, ETR or ETF)
0x2: ETF

Bit 5 **CLKSCHEM:** RAM clocking scheme (synchronous or asynchronous)
0x0: synchronous

Bits 4:0 **ATBINPORTCNT[4:0]**: number / type of ATB input port multiplexing
0x0: none

ETF device type identifier register (ETF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x3: captures trace data from the ATB slave interface into RAM that can be drained through the ATB master interface

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x2: component is a trace link because it has an ATB master interface through which trace data can be drained out in Hardware FIFO mode

ETF CoreSight peripheral identity register 4 (ETF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETF CoreSight peripheral identity register 0 (ETF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x61: ETF part number

ETF CoreSight Peripheral Identity register 1 (ETF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x9: ETF part number

ETF CoreSight peripheral identity register 2 (ETF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ETF CoreSight peripheral identity register 3 (ETF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ETF CoreSight component identity register 0 (ETF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

ETF CoreSight peripheral identity register 1 (ETF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: Common ID value

ETF CoreSight component identity register 2 (ETF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

ETF CoreSight component identity register 3 (ETF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Embedded trace FIFO registers map

Table 580. ETF register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETF_RSZ	Res.	RSZ[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x00C	ETF_STS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
	Reset value																												1	1	1	0	0
0x010	ETF_RRD	RRD[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x014	ETF_RRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRP[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ETF_RWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWP[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ETF_TRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x020	ETF_CTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
	Reset value																																0

Table 580. ETF register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	ETF_RWD	RWD[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x028	ETF_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
	Reset value																															0	0
0x02C	ETF_LBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x030	ETF_CBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETF_BUFWM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x300	ETF_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
	Reset value																															1	0
0x304	ETF_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	Reset value																		0	0	0		0	0	0		0	0	0			0	0
0x308	ETF_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[3:0]				
	Reset value																												0	1	0	1	0
0xFA0	ETF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	ETF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB0	ETF_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 580. ETF register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFB4	ETF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	ETF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	ETF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMWDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
	Reset value																						0	1	1	1	0	0	0	0	0	0	0
0xFD0	ETF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	1	1	0	0	1	0
0xFD0	ETF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	ETF_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	ETF_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	ETF_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	ETF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	0	0	0	0	1
0xFE4	ETF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1

Table 580. ETF register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE8	ETF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.6 Trace port interface unit (TPIU)

The TPIU is a CoreSight™ component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and sixteen data outputs, TRACED(15:0). The trace port width is programmable in the range 1 to 16. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes. However it restricts the bandwidth of the trace port and hence the quantity of trace information that can be output in real time.

For more information on the Trace Port Interface CoreSight™ component, refer to the *Arm® CoreSight™ SoC-400 Technical Reference Manual* [2].

Trace port interface unit registers

TPIU supported port size register (TPIU_SUPPSIZE)

Address offset: 0x000

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: supported trace port sizes
Indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.
0x0000 FFFF: port sizes 1 to 16 supported

TPIU current port size register (TPIU_CURPSIZE)

Address offset: 0x004

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PORTSIZE[31:0]**: current trace port size
Indicates current trace port size. Bit n-1 when set indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.

TPIU supported trigger modes register (TPIU_SUPTRGM)

Address offset: 0x100

Reset value: 0x0000 011F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
							r				r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **TRGRUN**: trigger running
0: trigger has not occurred or counter is at 0
1: trigger has occurred and counter is not at 0

Bit 16 **TRIGD**: triggered
0: trigger has not occurred
1: trigger has occurred and counter has reached 0

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TCOUNT8**: 8-bit counter register
1: implemented

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: multiplying the trigger counter by 65536 support
1: supported

Bit 3 **MULT256**: multiplying the trigger counter by 256 support
1: supported

Bit 2 **MULT16**: multiplying the trigger counter by 16 support
1: supported

Bit 1 **MULT4**: multiplying the trigger counter by 4 support
1: supported

Bit 0 **MULT2**: multiplying the trigger counter by 2 support
1: supported

TPIU trigger counter value register (TPIU_TRGCNT)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGCOUNT[7:0]**: trigger delay to external enable
Enables delaying the indication of triggers to any external connected trace capture or storage devices.
This counter is only eight-bit wide and is intended to be used only with the counter multipliers in the TPIU_TRGMULT trigger multiplier register. When a trigger is started, this value, in combination with the multiplier, is the number of words before the trigger is indicated. When the trigger counter reaches 0, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but does not reset any values on the multiplier. Reading this register returns the preset value, not the current count.

TPIU trigger multiplier register (TPIU_TRGMULT)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: multiply the trigger counter by 65536
0: disabled
1: enabled

Bit 3 **MULT256**: multiply the trigger counter by 256
0: disabled
1: enabled

Bit 2 **MULT16**: multiply the trigger counter by 16
0: disabled
1: enabled

Bit 1 **MULT4**: multiply the trigger counter by 4
0: disabled
1: enabled

Bit 0 **MULT2**: multiply the trigger counter by 2
0: disabled
1: enabled

TPIU supported test patterns/modes register (TPIU_SUPTPM)

Address offset: 0x200

Reset value: 0x0003 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: continuous mode support
Indicates whether continuous mode is supported.
1: supported

Bit 16 **PTIMEEN**: timed mode support
Indicates whether timed mode is supported.
1: supported

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: FF/00 pattern support
Indicates whether the FF/00 pattern is supported as output over the trace port.
1: supported

Bit 2 **PATA5**: AA/55 pattern support
Indicates whether the AA/55 pattern is supported as output over the trace port.
1: supported

Bit 1 **PATW0**: walking 0's pattern support
Indicates whether the walking 0's pattern is supported as output over the trace port.
1: supported

Bit 0 **PATW1**: walking 1's pattern support
Indicates whether the walking 1's pattern is supported as output over the trace port.
1: supported

TPIU current test pattern/mode register (TPIU_CURTPM)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: continuous mode enable
Indicates whether continuous mode is enabled.
0: disabled
1: enabled

Bit 16 **PTIMEEN**: timed mode enable
Indicates whether timed mode is enabled.
0: disabled
1: enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: FF/00 pattern enable
Indicates whether the FF/00 pattern is enabled as output over the trace port.
0: disabled
1: enabled

Bit 2 **PATA5**: AA/55 pattern enable
Indicates whether the AA/55 pattern is enabled as output over the trace port.
0: disabled
1: enabled

Bit 1 **PATW0**: walking 0's pattern enable
Indicates whether the walking 0's pattern is enabled as output over the trace port.
0: disabled
1: enabled

Bit 0 **PATW1**: walking 1's pattern enable
Indicates whether the walking 1's pattern is enabled as output over the trace port.
0: disabled
1: enabled

TPIU test pattern repeat counter register (TPIU_TPRCR)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PATTCOUNT[7:0]**: pattern number of TRACECLKIN cycles
8-bit counter value to indicate the number of TRACECLKIN cycles for which a pattern runs before it switches to the next pattern.

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TCPRESENT**: TRACECTL output pin is availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device

Bit 1 **FTSTOPPED**: formatter stopped

The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: formatter has not stopped

1: formatter has stopped

Bit 0 **FLINPROG**: flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.

0: no flush in progress

1: flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STOPTRIG	STOPFL	Res.	TRIGFL	TRIGEVT	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
		rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STOPTRIG**: stop on trigger event

0: no effect

1: stop formatter when a trigger event occurs

Bit 12 **STOPFL**: stop on flush

0: no effect

1: stop formatter when flush is completed

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGFL**: trigger on flush

0: no effect

1: indicates a trigger in the trace stream when flush is completed

Bit 9 **TRIGEVT**: trigger on trigger event

0: no effect

1: indicates a trigger in the trace stream when trigger event occurs

Bit 8 **TRIGIN**: trigger on trigger in
0: no effect
1: indicates a trigger in the trace stream when the TRIGIN input from the system CTI is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: generates a manual flush
0: no effect
1: flush the trace
This bit is cleared automatically when the flush completes.

Bit 5 **FONTRIG**: flush on trigger event
A trigger event occurs when the trigger counter reaches 0, or, if the trigger counter is 0, when the TRIGIN input from the system CTI is high.
0: no effect
1: flushes the trace if a trigger event occurs

Bit 4 **FONFLIN**: flush on flush in
0: no effect
1: flushes the trace if the FLUSHIN input from the system CTI is asserted

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT**: enables continuous formatting
0: continuous formatting is disabled
1: continuous formatting is enabled

Bit 0 **ENFTC**: enables the embedding of triggers in formatted trace
0: formatting is disabled
1: formatting is enabled

TPIU formatter synchronisation counter register (TPIU_FSCR)

Address offset: 0x308

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CYCCOUNT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **CYCCOUNT[4:0]**: formatted frame number

Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device.

This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 byte per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: sets claim tag bits

Write:
- 0000: no effect
- xxx1: sets bit 0
- xx1x: sets bit 1
- x1xx: sets bit 2
- 1xxx: sets bit 3

Read:
- 0xF: indicates there are four bits in claim tag

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: resets claim tag bits

Write:

– 0000: no effect
– xxx1: clears bit 0
– xx1x: clears bit 1
– x1xx: clears bit 2
– 1xxx: clears bit 3

Read:

– Returns current value of claim tag

TPIU lock access register (TPIU_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: TPIU register write access enable

Enables write access to some TPIU registers by processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: enable write access

Other values: disable write access

TPIU lock status register (TPIU_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: TPIU_LAR register size
Indicates the size of the TPIU_LAR register.
0: 32-bit

Bit 1 **LOCKGRANT**: current lock status
This bit always reads as zero by an external debugger.
0: write access is permitted
1: write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST**: lock control existence
Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.
0: no lock control mechanism exists
1: lock control mechanism is implemented

TPIU authentication status register (TPIU_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x0: not implemented

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x0: not implemented

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

TPIU device configuration register (TPIU_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial Wire Output, UART or NRZ support
Indicates whether Serial Wire Output, UART or NRZ, is supported.
0x0: not supported

Bit 10 **SWOMAN**: Serial Wire Output, Manchester support
Indicates whether Serial Wire Output, Manchester encoded format, is supported.
0x0: not supported

Bit 9 **TCLKDATA**: trace clock plus data support
Indicates whether trace clock plus data is supported.
0x0: not supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN sync
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous).
0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number / type of ATB input port multiplexing
0x0: none

TPIU device type identifier register (TPIU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x1: trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x12: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x9: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 004B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x4: r0p5

Bit 3 **JEDEC:** JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Trace port interface unit registers map

Table 581. TPIU register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TPIU_SUPPSIZE**	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x004	**TPIU_CURPSIZE**	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x100	**TPIU_SUPTRGM**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value															0	0								1				1	1	1	1	1
0x104	**TPIU_TRGCNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x108	**TPIU_TRGMULT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value																												0	0	0	0	0
0x200	**TPIU_SUPTPM**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value															1	1													1	1	1	1

Table 581. TPIU register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x204	TPIU_CURTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value															0	0													0	0	0	0
0x104	TPIU_TPRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																														0	1	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPTRIG	STOPFL	Res.	TRiGFL	TRiGEVt	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
	Reset value																			0	0		0	0	0		0	0	0			0	0
0x308	TPIU_FSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
	Reset value																					0	0	0	0	0	1	0	0	0	0	0	0
0xFA0	TPIU_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB0	TPIU_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	TPIU_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	TPIU_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0

Table 581. TPIU register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFC8	**TPIU_DEVID**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
	Reset value																					0	0	0	0	1	0	1	0	0	0	0	0
0xFD0	**TPIU_DEVTYPE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	**TPIU_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	**TPIU_PIDR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	**TPIU_PIDR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	**TPIU_PIDR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	**TPIU_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	1	0
0xFE4	**TPIU_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	**TPIU_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	1	0	0	1	0	1	1
0xFEC	**TPIU_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**TPIU_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1

Table 581. TPIU register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.7 Serial wire output (SWO)

The SWO is a CoreSight™ component that formats the trace stream from the Cortex®-M4 processor ITM and outputs it on the single wire TRACESWO output.

Compared to the TPIU, the SWO contains:

- no formatter
- no pattern generator
- an 8-bit ATB input
- no synchronous trace output (no TRACEDATA or TRACECLK pins)
- no support for flush
- no support for triggering

The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the Serial Wire Output CoreSight™ component, refer to the *Arm® CoreSight™ Components Technical Reference Manual [4]*.

Serial wire output registers

SWO current output divisor register (SWO_CODR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: 13-bit counter pre-load value

This register allows to scale the baud rate of the SWO output. The baud rate is the trace clock frequency divided by (PRESCALER - 1).

Note: The baud rate changes instantly; it is recommended to stop the trace source and wait until the port is idle before writing this register.

SWO selected pin protocol register (SWO_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **PPROT[1:0]**: pin protocol

0x0: reserved
0x1: Manchester
0x2: NRZ
0x3: reserved

SWO formatter and flush status register (SWO_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: no need to stop formatter to change settings

1: formatter can not be stopped

Bit 2 **TCPRESENT**: TRACECTL pin presence
0: TRACECTL pin is not present on SWO

Bit 1 **FTSTOPPED**: formatter stopped
0: formatter stopping is not supported

Bit 0 **FLINPROG**: flush in progress
0: flushing is not supported

SWO claim tag set register (SWO_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: sets claim tag bits
Write:
– 0000: no effect
– xxx1: sets bit 0
– xx1x: sets bit 1
– x1xx: sets bit 2
– 1xxx: sets bit 3
Read:
– 0xF: indicates there are four bits in claim tag

SWO claim tag clear register (SWO_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: resets claim tag bits

Write:
- 0000: no effect
- xxx1: clears bit 0
- xx1x: clears bit 1
- x1xx: clears bit 2
- 1xxx: clears bit 3

Read:
- Returns current value of claim tag

SWO lock access register (SWO_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: SWO register write access enable

Enables write access to some SWO registers by processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: enable write access

Other values: disable write access

SWO lock status register (SWO_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: SWO_LAR register size
Indicates the size of the SWO_LAR register.
0: 32-bit

Bit 1 **LOCKGRANT**: current lock status
This bit always reads as zero by an external debugger.
0: write access is permitted
1: write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST**: lock control existence
Indicates whether a lock control mechanism exists. This bit always reads as zero by an external debugger.
0: no lock control mechanism exists
1: lock control mechanism is implemented

SWO authentication status register (SWO_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x0: not implemented

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x0: not implemented

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

SWO device configuration register (SWO_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0EA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial Wire Output, UART or NRZ support

Indicates whether Serial Wire Output, UART or NRZ, is supported.

0x0: not supported

Bit 10 **SWOMAN**: Serial Wire Output, Manchester support

Indicates whether Serial Wire Output, Manchester encoded format, is supported.

0x0: not supported

Bit 9 **TCLKDATA**: trace clock plus data support

Indicates whether trace clock plus data is supported.

0x0: not supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2

0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT:** ATB clock and TRACECLKIN synchronization

Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous).

0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number / type of ATB input port multiplexing

0x0: none

SWO device type identifier register (SWO_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification

0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]**: major classification

0x1: trace sink component

SWO CoreSight peripheral identity register 4 (SWO_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

SWO CoreSight peripheral identity register 0 (SWO_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x14: SWO part number

SWO CoreSight peripheral identity register 1 (SWO_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x9: SWO part number

SWO CoreSight peripheral identity register 2 (SWO_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x4: r0p5

Bit 3 **JEDEC:** JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

SWO CoreSight peripheral identity register 3 (SWO_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

SWO CoreSight component identity register 0 (SWO_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

SWO CoreSight peripheral identity register 1 (SWO_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

SWO CoreSight component identity register 2 (SWO_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

SWO CoreSight component identity register 3 (SWO_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Serial wire output registers map

Table 582. SWO registers map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**SWO_SPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x004	**SWO_CPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x010	**SWO_CODR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F0	**SWO_SPPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
	Reset value																															0	1
0x100	**SWO_STMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x200	**SWO_STPR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Table 582. SWO registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	SWO_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	SWO_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFA0	SWO_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	SWO_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFB0	SWO_LAR	ACCESS_W[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xFB4	SWO_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	SWO_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	SWO_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
	Reset value																					1	1	1	0	1	0	1	0	0	0	0	0
0xFCC	SWO_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	0	1	0	0	0	1

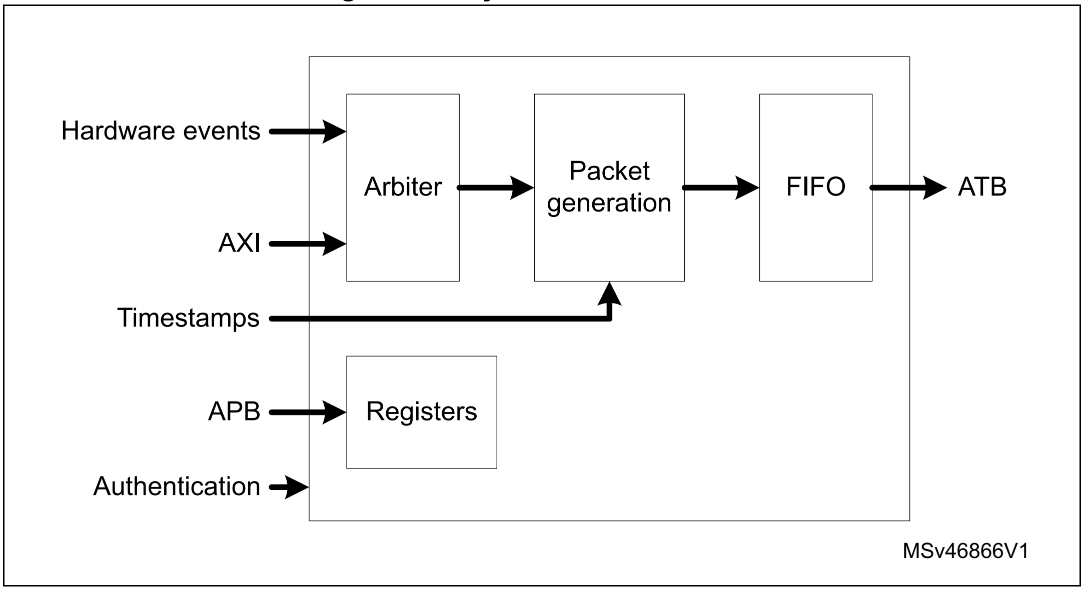
Table 582. SWO registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	SWO_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	SWO_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	SWO_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	SWO_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	SWO_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	1	0	0
0xFE4	SWO_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	SWO_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	SWO_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SWO_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SWO_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	SWO_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SWO_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.8 System trace macrocell (STM)

The System Trace Macrocell is a CoreSight™ component which allows instrumentation data generated by software to be incorporated in the trace output. The software writes the instrumentation data to the STM via a memory-mapped port on the AXI system interconnect. The data is formatted and time-stamped for output on the ATB trace bus, where it is merged with the trace coming from other sources (ETM, ITM).

Figure 793. System Trace Macrocell



The STM is configured with the following values:

- DATA_FIFO_DEPTH = 16
- CHN_FIFO_DEPTH = 8
- HW_EVOBIF_PRESENT = TRUE
- AXI_ID_WIDTH = 11

The various AXI masters can generate trace by writing to the extended stimulus ports of the STM via its AXI slave port, which is mapped in a 16-Mbyte window of address space (see memory map). A 256-byte extended stimulus port comprises 16 x 64-bit locations or registers. Writes to different locations generate different types of trace packets, as shown in *Table 583*. The base address for stimulus port n is the STM base address + n x 256. A master can use any number of the 65536 available stimulus ports mapped in the 16 Mbyte-window.

Table 583. STM extended stimulus port memory map

Address offset	Name	Description
0x00	G_DMTS	Data, marked with timestamp, guaranteed
0x08	G_DM	Data, marked, guaranteed
0x10	G_DTS	Data, with timestamp, guaranteed
0x18	G_D	Data, guaranteed
0x20-0x58	-	Reserved

Table 583. STM extended stimulus port memory map (continued)

Address offset	Name	Description
0x60	G_FLAGTS	Flag with timestamp, guaranteed
0x68	G_FLAG	Flag, guaranteed
0x70	G_TRIGTS	Trigger with timestamp, guaranteed
0x78	G_TRIG	Trigger, guaranteed
0x80	I_DMTS	Data, marked with timestamp, invariant timing
0x88	I_DM	Data, marked, invariant timing
0x90	I_DTS	Data, with timestamp, invariant timing
0x98	I_D	Data, invariant timing
0xA0 to 0xD8	-	Reserved
0xE0	I_FLAGTS	Flag with timestamp, invariant timing
0xE8	I_FLAG	Flag, invariant timing
0xF0	I_TRIGTS	Trigger with timestamp, invariant timing
0xF8	I_TRIG	Trigger, invariant timing

The trace packets contain the 8-bit identity of the master that triggered them. This identity is decoded from the non-secure master ID (NSAID) carried on the AWUSER wires of the AXI interconnect, together with the AWID signals which identify which of the two Cortex®-A7 CPU cores performed the write. The ID also includes the security status of the master at the time.

The ID is mapped as follows:

- Bit [7] = 0
- Bit [6] = secure/non-secure state
- Bit [5:0] = master ID

A list of valid IDs and how they map to the originating master is given in *Table 584*.

Table 584. STM trace packet ID mapping to AXI masters

Master	Secure access ID	Non-secure access ID
Cortex®-A7 CPU1	0x00	0x40
Cortex®-A7 CPU2	0x01	0x41
Cortex®-M4	0x02	0x42
MDMA	0x0A	0x4A
DMA1/2	-	0x4C
DAP	0x1E	0x5E

The STM can also generate trace packets or triggers when certain hardware events are detected. A maximum of 64 events can be traced at any one time. An external multiplexer allows four sets of signals to be mapped onto these events. The mapping is given in *Table 585* and *Table 586*. All events are edge sensitive.

Table 585. Hardware event mapping 0 to 3

Bank	Event	EXTMUX(7:0) = 0	EXTMUX(7:0) = 1	EXTMUX(7:0) = 2	EXTMUX(7:0) = 3
HEBS = 0	0	HWEVENTA↑	UART4_TX_sreq	WWDG1_EWIT_irq↑	EXTI6_irq↑
	1	HWEVENTA↓	UART5_RX_sreq	PVD_irq↑	EXTI7_irq↑
	2	HWEVENTB↑	UART5_TX_sreq	TAMP_STAMP_irq↑	EXTI8_irq↑
	3	HWEVENTB↓	DAC1_sreq	RTC_WKUP_irq↑	EXTI9_irq↑
	4	-	DAC2_sreq	-	GPDMA2_Stream5_irq↑
	5	-	TIM6_UP_sreq	RCC_irq↑	GPDMA2_Stream6_irq↑
	6	-	TIM7_UP_sreq	EXTI0_irq↑	GPDMA2_Stream7_irq↑
	7	-	USART6_RX_sreq	EXTI1_irq↑	USART6_irq↑
	8	-	USART6_TX_sreq	EXTI2_irq↑	I2C3_EVT_irq↑
	9	ADC1_sreq	I2C3RX_sreq	EXTI3_irq↑	I2C3_ERR_irq↑
	10	ADC2_sreq	I2C3TX_sreq	EXTI4_irq↑	USB_OHCI_IT_irq↑
	11	TIM1_CH1_sreq	DCMI_sreq	GPDMA1_Stream0_irq↑	USB_EHCI_IT_irq↑
	12	TIM1_CH2_sreq	CRYP2_IN_sreq	GPDMA1_Stream1_irq↑	EXTI12_irq↑
	13	TIM1_CH3_sreq	CRYP2_OUT_sreq	GPDMA1_Stream2_irq↑	EXTI13_irq↑
	14	TIM1_CH4_sreq	HASH2_IN_sreq	GPDMA1_Stream3_irq↑	DCMI_irq↑
	15	TIM1_UP_sreq	UART7_RX_sreq	GPDMA1_Stream4_irq↑	CRYP1_irq↑
	16	TIM1_TRIG_sreq	UART7_TX_sreq	GPDMA1_Stream5 _irq↑	HASH1_RNG1_irq↑
	17	TIM1_COM_sreq	UART8_RX_sreq	GPDMA1_Stream6_irq↑	FPU_irq↑
	18	TIM2_CH1_sreq	UART8_TX_sreq	ADC1_irq↑	UART7_irq↑
	19	TIM2_CH2_sreq	SPI4_RX_sreq	FDCAN1_IT0_irq↑	UART8_irq↑
	20	TIM2_CH3_sreq	SPI4_TX_sreq	FDCAN2_IT0_irq↑	SPI4_irq↑
	21	TIM2_CH4_sreq	SPI5_RX_sreq	FDCAN1_IT1_irq↑	SPI5_irq↑
	22	TIM2_UP_sreq	SPI5_TX_sreq	FDCAN2_IT1_irq↑	SPI6_irq↑
	23	TIM3_CH1_sreq	SAI1_A_sreq	EXTI5_irq↑	SAI1_irq↑
	24	TIM3_CH2_sreq	SAI1_B_sreq	TIM1_BRK_irq↑	LTDC_irq↑
	25	TIM3_CH3_sreq	SAI2_A_sreq	TIM1_UP_irq↑	LTDC_ER_irq↑
	26	TIM3_CH4_sreq	SAI2_B_sreq	TIM1_TRG_COM_irq↑	ADC2_irq↑
	27	TIM3_UP_sreq	DFSDM5_sreq	TIM1_CC_irq↑	SAI2_irq↑
	28	TIM3_TRIG_sreq	DFSDM6_sreq	TIM2_irq↑	QuadSPI_irq↑
	29	TIM4_CH1_sreq	SPDIFRX_DT_sreq	TIM3_irq↑	LPTIMER1_IT_irq↑
	30	TIM4_CH2_sreq	SPDIFRX_CS_sreq	TIM4_irq↑	CEC_irq↑
	31	TIM4_CH3_sreq	-	I2C1_EVT_irq↑	I2C4_EVT_irq↑

Table 585. Hardware event mapping 0 to 3 (continued)

Bank	Event	EXTMUX(7:0) = 0	EXTMUX(7:0) = 1	EXTMUX(7:0) = 2	EXTMUX(7:0) = 3
HEBS = 1	0	TIM4_UP_sreq	-	I2C1_ERR_irq↑	I2C4_ERR_irq↑
	1	I2C1_RX_sreq	-	I2C2_EVT_irq↑	SPDIFRX_irq↑
	2	I2C1_TX_sreq	-	I2C2_ERR_irq↑	OTG_IT_irq↑
	3	I2C2_RX_sreq	SAI4_A_sreq	SPI1_irq↑	-
	4	I2C2_TX_sreq	SAI4_B_sreq	SPI2_irq↑	IPCC_RX0_IT_irq↑
	5	SPI1_RX_sreq	DFSDM1_sreq	USART1_irq↑	IPCC_TX0_IT_irq↑
	6	SPI1_TX_sreq	DFSDM2_sreq	USART2_irq↑	DMA_MUX_OVR_REQ_IT_irq↑
	7	SPI2_RX_sreq	DFSDM3_sreq	USART3_irq↑	IPCC_RX1_IT_irq↑
	8	SPI2_TX_sreq	DFSDM4_sreq	EXTI10_irq↑	IPCC_TX1_IT_irq↑
	9	-	TIM15_CH1_sreq	RTC_Alarm_irq↑	CRYP2_irq↑
	10	-	TIM15_UP_sreq	EXTI11_irq↑	HASH2_RNG2_irq↑
	11	USART2_RX_sreq	TIM15_TRIG_sreq	TIM8_BRK_TIM12_irq↑	I2C5_EVT_irq↑
	12	USART2_TX_sreq	TIM15_COM_sreq	TIM8_UP_TIM13_irq↑	I2C5ER_irq↑
	13	USART3_RX_sreq	TIM16_CH1_sreq	TIM8_TRG_COM_TIM14_irq↑	-
	14	USART3_TX_sreq	TIM16_UP_sreq	TIM8_CC_irq↑	DFSDM1_IT_irq↑
	15	TIM8_CH1_sreq	TIM17_CH1_sreq	GPDMA1_Stream7_irq↑	DFSDM2_IT_irq↑
	16	TIM8_CH2_sreq	TIM17_UP_sreq	FMC_irq↑	DFSDM3_IT_irq↑
	17	TIM8_CH3_sreq	SAI3_A_sreq	SDMMC1_irq↑	DFSDM4_IT_irq↑
	18	TIM8_CH4_sreq	SAI3_B_sreq	TIM5_irq↑	SAI3_IT_irq↑
	19	TIM8_UP_sreq	I2C5_RX_sreq	SPI3_irq↑	-
	20	TIM8_TRIG_sreq	I2C5_TX_sreq	UART4_irq↑	TIM15_IT_irq↑
	21	TIM8_COM_sreq	USART1_RX_sreq	UART5_irq↑	TIM16_IT_irq↑
	22		USART1_TX_sreq	TIM6_DAC_irq↑	TIM17_IT_irq↑
	23	TIM5_CH1_sreq	SPI6_RX_sreq	TIM7_irq↑	MDIO_ASYNC_IT_irq↑
	24	TIM5_CH2_sreq	SPI6_TX_sreq	GPDMA2_Stream0_irq↑	MDIO_IT_irq↑
	25	TIM5_CH3_sreq	I2C4_RX_sreq	GPDMA2_Stream1_irq↑	EXTI14_irq↑
	26	TIM5_CH4_sreq	I2C4_TX_sreq	GPDMA2_Stream2_irq↑	MDMA_IT_irq↑
	27	TIM5_UP_sreq	I2C6_RX_sreq	GPDMA2_Stream3_irq↑	-
	28	TIM5_TRIG_sreq	I2C6_TX_sreq	GPDMA2_Stream4_irq↑	SDMMC2_irq↑
	29	SPI3_RX_sreq	-	ETH1_irq↑	HSEM_IT1_irq↑
	30	SPI3_TX_sreq	-	ETH1_WKUP_irq↑	HSEM_IT2_irq↑
	31	UART4_RX_sreq	-	FDCAN_CCU_irq↑	EXTI15_irq↑

Key: sreq = DMA request; irq = interrupt request;↑ = active going;↓ = inactive going

Table 586. Hardware event mapping 4 to 7

Bank	Event	EXTMUX(7:0) = 4	EXTMUX(7:0) = 5	EXTMUX(7:0) = 6	EXTMUX(7:0) = 7
HEBS = 0	0	nCTI1_irq↑	WWDG1_EWIT_irq↓	EXTI6_irq↓	nCTI1_irq↓
	1	nCTI2_irq↑	PVD_irq↓	EXTI7_irq↓	nCTI2_irq↓
	2	TIM13_irq↑	TAMP_STAMP_irq↓	EXTI8_irq↓	TIM13_irq↓
	3	TIM14_irq↑	RTC_WKUP_irq↓	EXTI9_irq↓	TIM14_irq↓
	4	DAC_irq↑	TZC_IT_irq↓	GPDMA2_Stream5_irq↓	DAC_irq↓
	5	RNG1_irq↑	RCC_irq↓	GPDMA2_Stream6_irq↓	RNG1_irq↓
	6	RNG2_irq↑	EXTI0_irq↓	GPDMA2_Stream7_irq↓	RNG2_irq↓
	7	I2C6_EVT_irq↑	EXTI1_irq↓	USART6_irq↓	I2C6_EVT_irq↓
	8	I2C6_ERR_irq↑	EXTI2_irq↓	I2C3_EVT_irq↓	I2C6_ERR_irq↓
	9	SDMMC3_irq↑	EXTI3_irq↓	I2C3_ERR_irq↓	SDMMC3_irq↓
	10	LPTIMER2_IT_irq↑	EXTI4_irq↓	USB_OHCI_IT_irq↓	LPTIMER2_IT_irq↓
	11	LPTIMER3_IT_irq↑	GPDMA1_Stream0_irq↓	USB_EHCI_IT_irq↓	LPTIMER3_IT_irq↓
	12	LPTIMER4_IT_irq↑	GPDMA1_Stream1_irq↓	EXTI12_irq↓	LPTIMER4_IT_irq↓
	13	LPTIMER5_IT_irq↑	GPDMA1_Stream2_irq↓	EXTI13_irq↓	LPTIMER5_IT_irq↓
	14	ETH1_LPI_irq ↑	GPDMA1_Stream3_irq↓	DCMI_irq↓	ETH1_LPI_irq ↓
	15	-	GPDMA1_Stream4_irq↓	CRYP1_irq↓	-
	16	MPU_SEV_irq↑	GPDMA1_Stream5_irq↓	HASH1_RNG1_irq↓	MPU_SEV_irq↓
	17	-	GPDMA1_Stream6_irq↓	FPU_irq↓	-
	18	SAI4_IT_irq↑	ADC1_irq↓	UART7_irq↓	SAI4_IT_irq↓
	19	DTS_IT_irq↑	FDCAN1_IT0_irq↓-	UART8_irq↓	DTS_IT_irq↓
	20	-	FDCAN2_IT0_irq↓	SPI4_irq↓	-
	21	WAKEUP_PIN_IT_irq↑	FDCAN1_IT1_irq↓	SPI5_irq↓	WAKEUP_PIN_IT_irq↓
	22	IWDG1_IT_irq ↑	FDCAN2_IT1_irq↓	SPI6_irq↓	IWDG1_IT_irq ↓
	23	IWDG2_IT_irq ↑	EXTI5_irq↓	SAI1_irq↓	IWDG2_IT_irq ↓
	24	-	TIM1_BRK_irq↓	LTDC_irq↓	-
	25	-	TIM1_UP_irq↓	LTDC_ER_irq↓	-
	26	-	TIM1_TRG_COM_irq↓	ADC2_irq↓	-
	27	-	TIM1_CC_irq↓	SAI2_irq↓	-
	28	-	TIM2_irq↓	QuadSPI_irq↓	-
	29	-	TIM3_irq↓	LPTIMER1_IT_irq↓	-
	30	-	TIM4_irq↓	CEC_irq↓	-
	31	-	I2C1_EVT_irq↓	I2C4_EVT_irq↓	-

Table 586. Hardware event mapping 4 to 7 (continued)

Bank	Event	EXTMUX(7:0) = 4	EXTMUX(7:0) = 5	EXTMUX(7:0) = 6	EXTMUX(7:0) = 7
HEBS = 1	0	-	I2C1_ERR_irq↓	I2C4_ERR_irq↓	-
	1	-	I2C2_EVT_irq↓	SPDIFRX_irq↓	-
	2	-	I2C2_ERR_irq↓	OTG_IT_irq↓	-
	3	-	SPI1_irq↓	-	-
	4	-	SPI2_irq↓	IPCC_RX0_IT_irq↓	-
	5	-	USART1_irq↓	IPCC_TX0_IT_irq↓	-
	6	-	USART2_irq↓	DMA_MUX_OVR_REQ_IT_irq↓	-
	7	-	USART3_irq↓	IPCC_RX1_IT_irq↓	-
	8	-	EXTI10_irq↓	IPCC_TX1_IT_irq↓	-
	9	-	RTC_Alarm_irq↓	CRYP2_irq↓	-
	10	-	EXTI11_irq↓	HASH2_RNG2_irq↓	-
	11	-	TIM8_BRK_TIM12_irq↓	I2C5_EVT_irq↓	-
	12	-	TIM8_UP_TIM13_irq↓	I2C5ER_irq↓	-
	13	-	TIM8_TRG_COM_TIM14_irq↓	-	-
	14	-	TIM8_CC_irq↓	DFSDM1_IT_irq↓	-
	15	-	GPDMA1_Stream7_irq↓	DFSDM2_IT_irq↓	-
	16	-	FMC_irq↓	DFSDM3_IT_irq↓	-
	17	-	SDMMC1_irq↓	DFSDM4_IT_irq↓	-
	18	-	TIM5_irq↓	SAI3_IT_irq↓	-
	19	-	SPI3_irq↓	-	-
	20	-	UART4_irq↓	TIM15_IT_irq↓	-
	21	-	UART5_irq↓	TIM16_IT_irq↓	-
	22	-	TIM6_DAC_irq↓	TIM17_IT_irq↓	-
	23	-	TIM7_irq↓	MDIO_ASYNC_IT_irq↓	-
	24	-	GPDMA2_Stream0_irq↓	MDIO_IT_irq↓	-
	25	-	GPDMA2_Stream1_irq↓	EXTI14_irq↓	-
	26	-	GPDMA2_Stream2_irq↓	MDMA_IT_irq↓	-
	27	-	GPDMA2_Stream3_irq↓	-	-
	28	-	GPDMA2_Stream4_irq↓	SDMMC2_irq↓	-
	29	-	ETH1_irq↓	HSEM_IT1_irq↓	-
	30	-	ETH1_WKUP_irq↓	HSEM_IT2_irq↓	-
	31	-	FDCAN_CCU_irq↓	EXTI15_irq↓	-
Key: sreq = DMA request; irq = interrupt request;↑ = active going;↓ = inactive going					

For more information on the System Trace Macrocell, refer to the *Arm® CoreSight™ STM-500 Technical Reference Manual* [8].

STM registers

The STM is configured and controlled by registers mapped on the debug APB, accessed by the debugger at STMBaseAddress = 0xE00A 0000, and by software at STMBaseAddress = 0x500A 0000.

System trace macrocell registers

STM hardware event enable register (STM_HEER)

Address offset: 0xD00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HEE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HEE[31:0]**: hardware event enable
HEE is used to enable hardware events to generate trace. The register defines one bit per hardware event.
0: hardware event disabled
1: hardware event enabled

STM hardware event trigger enable register (STM_HETER)

Address offset: 0xD20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HETE[31:0]**: trigger generation enable
HETE is a bit mask used to enable trigger generation on hardware events. The register defines one bit per hardware event.
0: disabled
1: enabled

STM hardware event bank select register (STM_HEBSR)

Address offset: 0xD60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HEBS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **HEBS**: hardware event bank selector

HEBS is used to select the bank of 32 hardware events to control, see *Table 585: Hardware event mapping 0 to 3*).

0: bank 0

1: bank 1

STM hardware event main control register (STM_HEMCR)

Address offset: 0xD64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN	TRIGCLEAR	TRIGSTATUS	TRIGCTL	Res.	ERRDETECT	COMPEN	EN
								rw	rw	r	rw		r	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATBTRIGEN**: ATB trace trigger enable

When set, this bit enables the STM to use the ATID value of 0x7D when a trigger event on match using STMHETER occurs.

0: do not output ATB trace triggers

1: use ATB trace triggers

Bit 6 **TRIGCLEAR**: clear trigger status

When TRIGCTL indicates single-shot mode, this bit is used to clear TRIGSTATUS. Writing a 1 to this bit when in multi-shot mode is unpredictable.

0: no effect

1: clears TRIGSTATUS if TRIGSTATUS is 1

Bit 5 **TRIGSTATUS**: trigger status

When TRIGCTL indicates single-shot mode, this indicates whether the single trigger has occurred.

0: trigger has not occurred

1: trigger has occurred

Bit 4 **TRIGCTL**: trigger control

0: triggers are multi-shot

1: triggers are single-shot

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ERRDETECT**: error detection
Enable error detection on the hardware event tracing.
0: error detection disabled
1: error detection enabled.

Bit 1 **COMPEN**: compression enable
Enable leading zero suppression of hardware event data values in the trace stream.
0: compression disabled
1: compression enabled.

Bit 0 **EN**: enable hardware event tracing
0: disabled
1: enabled

STM hardware event external multiplex control register (STM_HEEXTMUXR)

Address offset: 0xD68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTMUX[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXTMUX[7:0]**: signals for events selector
Selects the signals to be mapped onto the 64 hardware events that can be traced by the STM (refer to *Table 585* and *Table 586*).

STM hardware event master number register (STM_HEMASTR)

Address offset: 0xDF4

Reset value: 0x0000 0080 (0x0000 FFFF)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MASTER[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MASTER[15:0]**: STPv2 master number

Indicates the STPv2 master number of hardware event trace. This number is the master number presented in STPv2.

0x80: hardware events are associated with master 0x80

Others: reserved

STM hardware event features 1 register (STM_HEFEAT1R)

Address offset: 0xDF8

Reset value: 0x3020 0035

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HEEXTMUXSIZE[2:0]			Res.	Res.	Res.	Res.	NUMHE[8:1]							
	r	r	r					r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NUMHE[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HECOMP[1:0]		HEMASTR	HEERR	Res.	HETER
r										r	r	r	r		r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **HEEXTMUXSIZE[2:0]**: EXTMUX bit field size

Indicates the size of the EXTMUX bit field of register STM_HEEXTMUXR.

0x3: 8-bit wide (up to 256 multiplexers)

Others: reserved

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:15 **NUMHE[8:0]**: number of hardware events supported by the STM

0x40: 64 hardware events

Bits 14:6 Reserved, must be kept at reset value.

Bits 5:4 **HECOMP[1:0]**: data compression on hardware event tracing support

0x3: data compression support is programmable. The COMPEN bit of register STM_HEMCR is implemented.

Bit 3 **HEMASTR**: STMHEMASTR support

0: STMHEMASTR is read-only

Bit 2 **HEERR**: hardware event error detection support.

1: hardware event error detection implemented. The ERRDETECT bit of register STM_HEMCR is implemented.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HETER**: STM_HETER support

1: STM_HETER implemented.

STM hardware event features ID register (STM_HEIDR)

Address offset: 0xDFC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	VENDSPEC[3:0]				CLASSREV[3:0]				CLASS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **VENDSPEC[3:0]**: vendor specific
Identifies any vendor specific modifications or mappings.
0: vendor specific information
Others: reserved

Bits 7:4 **CLASSREV[3:0]**: programmer model revision
Identifies the revision of the programmer model.
1: revision
Others: reserved

Bits 3:0 **CLASS[3:0]**: programmer model
Identifies the programmers model.
1: hardware event control
Others: reserved

STM stimulus port enable register (STM_SPER)

Address offset: 0xE00

Reset value: 0x0000 0000

This read/write only register is used to enable the stimulus registers to generate trace. The register defines one bit per stimulus register. Writing 1 enables the appropriate stimulus port, writing 0 disables the appropriate stimulus port. This register is used in conjunction with the Software Enable Bank Select Register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPE[31:0]**: stimulus port enable
This register is used to enable the stimulus registers to generate trace. It defines one bit per stimulus register. Writing 1 enables the appropriate stimulus port, writing 0 disables the appropriate stimulus port. This register is used in conjunction with the Software Enable Bank Select Register.
0: stimulus port disabled
1: stimulus port enabled

STM stimulus port trigger enable register (STM_SPTER)

Address offset: 0xE20

Reset value: 0x0000 0000

This register is used to enable trigger generation on writes to enabled stimulus port registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPTE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPTE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPTE[31:0]**: trigger generation trigger generation bit mask

This register is a bit mask used to enable trigger generation on writes to enabled stimulus port registers. It defines one bit per stimulus port register.

0: disabled

1: enabled

STM stimulus port select configuration register (STM_SPSCR)

Address offset: 0xE60

Reset value: 0x0000 0000

This register allows a debugger to program which stimulus ports the STM_SPER and STM_SPTER registers apply to.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSEL[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTCTL[1:0]	
														rw	rw

Bits 31:20 **PORTSEL[11:0]**: stimulus port selector

Defines which stimulus ports the STM_SPTER and/or STM_SPER registers apply to.

Bits 19:2 Reserved, must be kept at reset value.

Bits 1:0 **PORTCTL[1:0]**: port selection control mode

Defines how the port selection is applied.

0x0: port selection not used

0x1: port selection applies only to the STM_SPTER register

0x2: reserved

0x3: port selection applies to both the STM_SPER and STM_SPTER

STM stimulus port master select configuration register (STM_SPMSCR)

Address offset: 0xE64

Reset value: 0x0000 0000

This register allows a debugger to program which masters the STM_SPSCR register applies to.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:1]						
									rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MASTSEL[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
rw															rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:15 **MASTSEL[7:0]**:master select

This field defines which master the STM_SPSCR register applies to.

Bits 14:1 Reserved, must be kept at reset value.

Bit 0 **MASTCTL**: master control mode

Defines how the master is applied.

0: master selection not used

1: port selection applies to the STM_SPSCR register

STM stimulus port override control register (STM_SPOVERRIDER)

Address offset: 0xE68

Reset value: 0x0000 0000

This register allows a debugger to override various features of the STM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSEL[16:1]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PORTSEL[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVERTS	OVERCTL[1:0]	
rw													rw	rw	rw

Bits 31:15 **PORTSEL[16:0]**: port override select

Defines which stimulus ports the override controls apply to.

Bits 14:3 Reserved, must be kept at reset value.

Bit 2 **OVERTS**: override timestamp request

This override requests all stimulus port writes that cause trace to be traced with a timestamp (when possible). As with normal operation, this does not ensure that all packets are generated with timestamps. This field is independent from OVERCTL and PORTSEL.

0: override not enabled

1: override enabled

Bits 1:0 **OVERCTL[1:0]**: override control mode

Defines how port selection is applied.

0x0: override controls disabled

0x1: ports selected by PORTSEL always behave as guaranteed transactions

0x2: ports selected by PORTSEL always behave as invariant timing transactions

0x3: reserved

STM stimulus port master override control register (STM_SPMOVERRIDER)

Address offset: 0xE6C

Reset value: 0x0000 0000

This register allows a debugger to select which masters the STM_SPOVERRIDERR applies to.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:1]						
									rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MASTSEL[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
rw															rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:15 **MASTSEL[7:0]**: master port override selection

This field defines which master ports the override controls apply to. The size of this field is defined by the number of implemented masters.

Bits 14:1 Reserved, must be kept at reset value.

Bit 0 **MASTCTL**: master selection control mode

Defines how the master selection is applied.

0: override controls disabled. STM_SPOVERRIDERR applies equally to all masters.

1: master selection enabled. STM_SPOVERRIDERR applies to the masters selected by MASTSEL.

STM stimulus port trigger control and status register (STM_SPTRIGCSR)

Address offset: 0xE70

Reset value: 0x0000 0000

This register is used to control the STM triggers induced by STM_SPTER.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN_DIR	ATBTRIGEN_TE	TRIGCLEAR	TRIGSTATUS	TRIGCTL
											rw	rw	w	r	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ATBTRIGEN_DIR**: ATB trace on write to TRIG

When set, this bit enables the STM to use the ATID value of 0x7D when a trigger event on writes to TRIG location occurs.

0: does not output ATB trace triggers on write to TRIG location

1: outputs ATB trace trigger for trigger on write to TRIG location

Bit 3 **ATBTRIGEN_TE**: ATB trace trigger output

When set, this bit enables the STM to use the ATID value of 0x7D when a trigger event on match using STM_SPTER occurs.

0: does not output ATB trace trigger for trigger on match using STM_SPTER

1: outputs ATB trace trigger for trigger on match using STM_SPTER

Bit 2 **TRIGCLEAR**: trigger status clear

When bit TRIGCTL indicates single-shot mode, TRIGCLEAR is used to clear TRIGSTATUS. Writing a 1 to this bit when in multi-shot mode is leads to unpredictable behavior.

0: no effect

1: clears TRIGSTATUS if TRIGSTATUS is 1

Bit 1 **TRIGSTATUS**: trigger status

When bit TRIGCTL indicates single-shot mode, TRIGSTATUS indicates whether the single trigger has occurred.

0: trigger has not occurred

1: trigger has occurred

Bit 0 **TRIGCTL**: trigger control

0: triggers are multi-shot

1: triggers are single-shot

STM trace control and status register (STM_TCSR)

Address offset: 0xE80

Reset value: 0x0000 0004

Controls STM settings.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPEN	Res.	Res.	SYNCEN	TSEN	EN
										rw			r	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: STM busy

Indicates if the STM is busy, for example because STM trace FIFO is not empty.

0: STM is not busy

1: STM is busy.

Bits 22:16 **TRACEID[6:0]**: ATB Trace ID

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **COMPEN**: compression enable for stimulus ports

0: compression disabled, data transfers are transmitted at the size of the transaction

1: compression enabled, data transfers are compressed to save bandwidth

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SYNCEN**: STM_SYNCR implementation

STM_SYNCR is implemented; this value is RAO.

1: STM_SYNCR implemented

Bit 1 **TSEN**: timestamp requests enable

This bit controls if timestamp requests are ignored or taken into account.

0: timestamping disabled. Requests for timestamp generation are ignored, and stimulus port writes selecting timestamping are treated as if it were not selected.

1: timestamping enabled. If stimulus writes select timestamping, a timestamp is output according to STPv2.

Bit 0 **EN**: global STM enable

0: STM disabled

1: STM enabled

STM timestamp stimulus register (STM_TSSTIMR)

Address offset: 0xE84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCETS
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **FORCETS**: force timestamp stimulus

A write to register STM_TSSTIMR with bit FORCETS as 1 requests the next stimulus port write which causes trace to be upgraded to have a timestamp. Writes with this bit as 0 are ignored.

1: force timestamp stimulus

STM timestamp frequency register (STM_TSFREQR)

Address offset: 0xE8C

Reset value: 0x0000 0000

This register is used to indicate the frequency of the timestamp counter. The unit of measurement is increments per second. When the STPv2 protocol is used, this register contains the value output in the FREQ and FREQ_TS packets. The timestamp frequency is output in the STPv2 protocol at every synchronization point when bit TSEN STM_TCSR is 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]**: timestamp frequency

Indicates the frequency of the timestamp counter in Hz.

STM synchronization control register (STM_SYNCR)

Address offset: 0xE90

Reset value: 0x0000 0000

This register controls the interval between synchronization packets as a function of the number of bytes of trace generated. This register only provides a hint of the desired synchronization frequency since implementations are permitted to be inaccurate. Writing a value of 0x0000 0000 to this register disables the synchronization counter however any other implementation defined synchronization mechanism continues to operate independently (refer to register *STM auxiliary control register (STM_AUXCR)*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	MODE	COUNT[8:0]									Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **MODE**: mode control
0: COUNT[8:0] defines a value N. Synchronization period is 8 x N bytes.
1: COUNT[8:4] defines a value N. Synchronization period is 2^N bytes. N must be in the range of 0d12 to 0d27 inclusive; other values lead to unpredictable results.

Bits 11:3 **COUNT[8:0]**: number of bytes between synchronization packets
Indicates the counter value for the number of bytes between synchronization packets. Reads return the value of this register.

Bits 2:0 Reserved, must be kept at reset value.

STM auxiliary control register (STM_AUXCR)

Address offset: 0xE94

Reset value: 0x0000 0000

This register is used for implementation defined STM controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QHWEVOVERRIDE	Res.	Res.	Res.	Res.	PRIORINVDIS	ASYNCPE	FIFOAF
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **QHWEVOVERRIDE**: override control for the Q-Channels
0: no override. Q-Channel is able to accept a quiescence request when hardware event tracing is enabled.
1: override. Q-Channel denies a quiescence request when hardware event tracing is enabled.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **PRIORINVDIS**: AXI vs. HW flush arbitration
Controls arbitration between AXI and HW during flush.
0: priority inversion. When AXI flush is finished, HW gets priority until HW flush is done.
1: priority inversion disabled. AXI always has priority over HW.

Bit 1 **ASYNCPE**: ASYNC priority
0: ASYNC priority is always lower than trace
1: ASYNC priority increases on second synchronization request

Bit 0 **FIFOAF**: auto-flush
0: auto-flush disabled
1: auto-flush enabled

STM features 1 register (STM_FEAT1R)

Address offset: 0xEA0

Reset value: 0x0065 87D1

This register indicates the features of the STM in conjunction with registers STM_FEAT2R and STM_FEAT3R.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOEN[1:0]		SYNCEN[1:0]		HWTEN[1:0]		TSPRESCALE[1:0]	
								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGCTL[1:0]		TRACEBUS[3:0]				SYNC[1:0]		FORCETS	TSFREQ	TS[1:0]		PROT[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **SWOEN[1:0]**: SWOEN support
Support of bit SWOEN of register STM_TCSR.
0x1: bit SWOEN of register STM_TCSR is not implemented

Bits 21:20 **SYNCEN[1:0]**: SYNCEN support
Support of bit SYNCEN of register STM_TCSR.
0x2: bit SYNCEN of register STM_TCSR is implemented but always reads as 1

Bits 19:18 **HWTEN[1:0]**: HWTEN support
Support of bit HWTEN of register STM_TCSR.
0x1: bit HWTEN of register STM_TCSR is not implemented

Bits 17:16 **TSPRESCALE[1:0]**: timestamp prescale support
0x1: timestamp prescale not implemented

Bits 15:14 **TRIGCTL[1:0]**: trigger control support
0x2: multi-shot and single-shot triggers supported. STM_TRIGCSR is implemented.

Bits 13:10 **TRACEBUS[3:0]**: trace bus support
0x1: CoreSight™ ATB plus ATB trigger support implemented. Field TRACEID of register STM_TCSR, and bits ATBTRIGEN_DIR and ATBTRIGEN_TE of register STM_SPTRIGCSR are implemented.

Bits 9:8 **SYNC[1:0]**: STM_SYNCR support
0x3: STM_SYNCR implemented with MODE control

Bit 7 **FORCETS**: STM_TSSTIMR support
1: bit FORCETS of register STM_TSSTIMR is implemented

Bit 6 **TSFREQ**: Timestamp frequency indication configuration
1: STM_TSFREQR is read-write

Bits 5:4 **TS[1:0]**: Timestamp support
0x1: absolute timestamps implemented

Bits 3:0 **PROT[3:0]**: Protocol
0x1: STPv2 protocol

STM features 2 register (STM_FEAT2R)

Address offset: 0xEA4

Reset value: 0x0001 14F2

This register indicates the features of the STM in conjunction with registers STM_FEAT1R and STM_FEAT3R.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPTYPE[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSIZE[3:0]				Res.	SPTRTYPE[1:0]		PRIVMASK[1:0]		SPOVERRIDE	SPCOMP[1:0]		Res.	SPER	SPTER[1:0]	
r	r	r	r		r	r	r	r	r	r	r		r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **SPTYPE[1:0]**: stimulus port type support
0x1: only extended stimulus ports implemented

Bits 15:12 **DSIZE[3:0]**: fundamental data size
0x1: 64-bit data

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **SPTRTYPE[1:0]**: stimulus port transaction type support
0x2: both invariant timing and guaranteed transactions are supported

Bits 8:7 **PRIVMASK[1:0]**: STM_PRIVMASKR support
0x1: STM_PRIVMASKR is not implemented

Bit 6 **SPOVERRIDE**: STM_OVERRIDER support
1: STM_SPOVERRIDER and STM_SPMOVERRIDER are implemented

Bits 5:4 **SPCOMP[1:0]**: data compression on stimulus ports support
0x3: data compression support is programmable. Bit COMPEN of register STM_TCSR is implemented.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SPER**: STM_SPER presence
0: STM_SPER is implemented

Bits 1:0 **SPTER[1:0]**: STM_SPTER support
0x2: STM_SPTER is implemented

STM features 3 register (STM_FEAT3R)

Address offset: 0xEA8

Reset value: 0x0000 007F

This register indicates the features of the STM in conjunction with registers STM_FEAT1R and STM_FEAT2R.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMMAST[6:0]						
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **NUMMAST[6:0]**: number of stimulus ports masters implemented, minus 1
0x7F: 128 masters are implemented

STM integration test for cross-trigger outputs register (STM_ITTRIGGER)

Address offset: 0xEE8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASYNCOUT_W	TRIGOUTHETE_W	TRIGOUTSW_W	TRIGOUTSPTE_W
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ASYNCOUT_W**: sets the value of the ASYNCOUT output in integration mode
1: drives logic 1 on ASYNCOUT output
0: drives logic 0 on ASYNCOUT output

Bit 2 **TRIGOUTHETE_W**: sets the value of the TRIGOUTHETE output in integration mode
1: drives logic 1 on TRIGOUTHETE output
0: drives logic 0 on TRIGOUTHETE output

Bit 1 **TRIGOUTSW_W**: sets the value of the TRIGOUTSW output in integration mode
1: drives logic 1 on TRIGOUTSW output
0: drive logic 0 on TRIGOUTSW output

Bit 0 **TRIGOUTSPTE_W**: sets the value of the TRIGOUTSPTE output in integration mode
1: drives logic 1 on TRIGOUTSPTE output
0: drives logic 0 on TRIGOUTSPTE output

STM integration mode ATB data 0 register (STM_ITATBDATA0)

Address offset: 0xEEC

Reset value: 0x0000 0000

This register controls the value of the ATDATAM output in integration mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATDATAM63_W	ATDATAM55_W	ATDATAM47_W	ATDATAM39_W	ATDATAM31_W	ATDATAM23_W	ATDATAM15_W	ATDATAM7_W	ATDATAM0_W
							w	w	w	w	w	w	w	w	w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **ATDATAM63_W**: sets the value of the ATDATAM[63] output
1: drives logic 1 on ATDATAM[63] output
0: drives logic 0 on ATDATAM[63] output

Bit 7 **ATDATAM55_W**: sets the value of the ATDATAM[55] output
1: drives logic 1 on ATDATAM[55] output
0: drives logic 0 on ATDATAM[55] output

Bit 6 **ATDATAM47_W**: sets the value of the ATDATAM[47] output
1: drives logic 1 on ATDATAM[47] output
0: drives logic 0 on ATDATAM[47] output

Bit 5 **ATDATAM39_W**: sets the value of the ATDATAM[39] output
1: drives logic 1 on ATDATAM[39] output
0: drives logic 0 on ATDATAM[39] output

Bit 4 **ATDATAM31_W**: sets the value of the ATDATAM[31] output
1: drives logic 1 on ATDATAM[31] output
0: drives logic 0 on ATDATAM[31] output

Bit 3 **ATDATAM23_W**: sets the value of the ATDATAM[23] output
1: drives logic 1 on ATDATAM[23] output
0: drives logic 0 on ATDATAM[23] output

Bit 2 **ATDATAM15_W**: sets the value of the ATDATAM[15] output
1: drives logic 1 on ATDATAM[15] output
0: drives logic 0 on ATDATAM[15] output

Bit 1 **ATDATAM7_W**: sets the value of the ATDATAM[7] output
1: drives logic 1 on ATDATAM[7] output
0: drives logic 0 on ATDATAM[7] output

Bit 0 **ATDATAM0_W**: sets the value of the ATDATAM[0] output
1: drives logic 1 on ATDATAM[0] output
0: drives logic 0 on ATDATAM[0] output

STM integration mode ATB control 2 register (STM_ITATBCTR2)

Address offset: 0xEF0

Reset value: 0x0000 0000

This register returns the value of the ATREADYM and AFVALIDM inputs in integration mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFVALIDM_R	ATREADYM_R
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **AFVALIDM_R**: reads the value of the AFVALIDM input
1: pin is at logic 1
0: pin is at logic 0

Bit 0 **ATREADYM_R**: reads the value of the ATREADYM input
1: pin is at logic 1
0: pin is at logic 0

STM integration mode ATB identification register (STM_ITATBID)

Address offset: 0xEF4

Reset value: 0x0000 0000

This register controls the value of the ATIDM output in integration mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATIDM_W[6:0]						
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **ATIDM_W[6:0]**: sets the value of the ATIDM output

STM integration mode ATB control 0 register (STM_ITATBCTR0)

Address offset: 0xEF8

Reset value: 0x0000 0000

This register controls the value of the ATVALIDM, AFREADYM, and ATBYTESM outputs in integration mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ATBYTESM_W[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	AFREADYM_W	ATVALIDM_W
					w	w	w							w	w

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **ATBYTESM_W[2:0]**: sets the value of the ATBYTESM output
0x7: drives logic 111 on the ATBYTESM output
0x6: drives logic 110 on the ATBYTESM output
0x5: drives logic 101 on the ATBYTESM output
0x4: drives logic 100 on the ATBYTESM output
0x3: drives logic 11 on the ATBYTESM output
0x2: drives logic 10 on the ATBYTESM output
0x1: drives logic 01 on the ATBYTESM output
0x0: drives logic 00 on the ATBYTESM output

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **AFREADYM_W**: sets the value of the AFREADYM output
1: drives logic 1 on the AFREADYM output
0: drives logic 0 on the AFREADYM output

Bit 0 **ATVALIDM_W**: sets the value of the ATVALIDM output
1: drives logic 1 on the ATVALIDM output
0: drives logic 0 on the ATVALIDM output

STM integration mode control register (STM_ITCTRL)

Address offset: 0xF00

Reset value: 0x0000 0000

This register is used to enable topology detection. Refer to *CoreSight™ Architecture Specification* for more information. This register enables the component to switch between functional mode and integration mode. The default behavior is functional mode. In integration mode the inputs and outputs of the STM can be directly controlled for integration testing and topology solving.

Note: When a device has been in integration mode, it might not function with the original behavior. After performing integration or topology detection, the system must be reset to ensure correct behavior of CoreSight™ and other connected system components that are affected by the integration or topology detection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IME
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **IME**: component mode switch enable
Enables the component to switch between functional mode and integration mode.
1: enables integration mode
0: disables integration mode

STM claim tag set register (STM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

This register is used in conjunction with the Claim Tag Clear register (STM_CLAIMCLR). This register forms one half of the claim tag value. This location allows individual bits to be set (write) and returns the number of bits that can be set (read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SET[3:0]**: claim tag bit implementation
0xF: claim tag bits are present within the claim tag field

STM claim tag clear register (STM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

This register is used in conjunction with the Claim Tag Set register (STM_CLAIMSET). This register forms one half of the claim tag value. This location enables individual bits to be cleared (write) and returns the current claim tag value (read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLR[3:0]**: claim tag current setting

STM lock access register (STM_LAR)

Address offset: 0xFB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: device register write access enable

A write of 0xC5ACCE55 enables further write access to the device. An invalid write has the effect of removing write access.

0xC5ACCE55: unlocks the protection register to enable write access

STM lock status register (STM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

This register indicates the status of the lock control mechanism that prevents accidental writes by the code under debug. The lock mechanism does not impact accesses to the extended stimulus port registers. This register is needed even in the absence of any lock access control mechanism. Where it is present and locked, the lock mechanism blocks the write accesses to any register but the STM_LAR. The lock mechanism is only present for accesses with PADDRDBG31 signal low.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **NTT**: STM_LAR size indicator
Indicates if the STM implements the STM_LAR as 8-bit or 32-bit.
0: 32-bit

Bit 1 **SLK**: lock current status
0: enables write access to the STM. For read accesses to the STM_LSR when the PADDRDBG31 signal is high, this bit is always 0.
1: blocks write access to the STM. The STM ignores all write accesses to the registers when the PADDRDBG31 signal is low. Reads are permitted.

Bit 0 **SLI**: lock control mechanism existence
Indicates that a lock control mechanism exists for this device.
0: no lock control mechanism exists. The STM ignores writes to the STM_LAR. For read accesses to the STM_LSR when the PADDRDBG31 signal is high, this bit is always 0.
1: lock control mechanism is present. For read accesses to the STM_LSR when the PADDRDBG31 signal is low, this bit is always 1.

STM authentication status register (STM_AUTHSTATUS)

Address offset: 0xFB8

Reset value: 0x0000 00AA

This register reports the required security level and current status of the authentication interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x2: disabled
0x3: enabled

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x2: disabled
0x3: enabled

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x2: disabled
0x3: enabled

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x2: disabled
0x3: enabled

STM device architecture register (STM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4771 0A63

This register indicates the architect and architecture of the STM. For the STM-500, the architect is Arm®, and the architecture is STMv1.1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	ARCHID[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: component architect

Defines the architect of the component. Bits[31:28] indicate the DEP106 continuation code of the architect. Bits[27:21] indicate the JEP106 identification code of the architect. Refer to the Standard Manufacturers Identification Code for information about JEP106.

0x23B: the architect of the STM-500 is Arm®

Bit 20 **PRESENT**: STM_DEVARCH register presence

1: the STM_DEVARCH register is present

Bits 19:16 **REVISION[3:0]**: architecture revision

Returns the revision of the architecture that the ARCHID field specifies. For the STM-500, this value is 0x1, indicating the STMv1.1 architecture.

0x1: STMv1.1 architecture

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **ARCHID[14:0]**: architecture ID

Returns a value that identifies the architecture of the component. For the STM-500, this value is 0x0A63, indicating the STMv1 architecture.

0x0A63: STMv1 architecture

STM device configuration register (STM_DEVID)

Address offset: 0xFC8

Reset value: 0x0001 0000

This register indicates the capabilities of the CoreSight™ STM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMSP[16]
															r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NUMSP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **NUMSP[16:0]**: number of stimulus ports implemented

0x10000: 65 536 stimulus ports are implemented

STM device type identifier register (STM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0063

This register provides a debugger with component information when the part number is not recognized. The debugger can then report this information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-classification within the major category
0x6: CoreSight™ STM. The component generates trace based on software and hardware stimulus.

Bits 3:0 **MAJOR[3:0]**: major classification for debug or trace component
0x3: the component is a trace source

STM peripheral ID4 register (STM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

This register belongs to the set of peripheral identification registers. It contains part of the designer identity and the memory footprint indicator.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: component memory size indicator
Indicates the total contiguous size of the memory window used by the component in powers of two from the standard 4 Kbytes. If a component only requires the standard 4 Kbytes, this bit field must read as 0x0. For 8 Kbytes it is set to 0x1. For 16 Kbytes it set to 0x2. For 32 Kbytes it is set to 0x3, and similarly for larger memory windows.
0x0: the device only occupies 4 Kbytes of memory

Bits 3:0 **DES_2[3:0]**: JEDEC continuation code
Indicates the designer of the component, together with the identity code.
0x4: indicates that Arm® JEDEC identity code is on the 5th bank

STM peripheral ID0 register (STM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0063 (0x0000 00FF)

This register belongs to the set of peripheral identification registers. It contains part of the designer specific part number.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PART_0[7:0]**: bits [7:0] of the component part number
This field is specified by the designer of the component.
0x63: CoreSight™ STM Part Number[7:0]. Eight least significant bits of the part number (0x963).

STM peripheral ID1 register (STM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

This register belongs to the set of peripheral identification registers. It contains part of the designer specific part number and part of the designer identity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DES_0[3:0]**: bits [3:0] of the JEDEC identity code
Indicates the designer of the component, together with the continuation code.
0xB: least significant four bits of the JEP106 identity code.

Bits 3:0 **PART_1[3:0]**: bits [11:8] of the component part number
Specified by the designer of the component.
0x9: CoreSight™ STM part number[11:8]. Most significant four bits of the part number (0x963).

STM peripheral ID2 register (STM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

This register belongs to the set of peripheral identification registers. It contains part of the designer identity and the product revision.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: incremental component design version

An incremental value starting from 0x0 for the first design of this component. The value increases by one at both major and minor revisions and is used as a look-up to establish the exact major and minor revisions.

0x1: this device is at r0p1.

Bit 3 **JEDEC**: JEDEC assigned value usage

Indicates the use of a JEDEC assigned value. This bit is always set.

1: the designer ID is specified by JEDEC (refer to http://www.jedec.org)

Bits 2:0 **DES_1[2:0]**: bits [6:4] of the JEDEC identity code

Indicates the designer of the component, together with the continuation code.

0x3: most significant three bits of the JEP106 identity code

STM peripheral ID3 register (STM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register belongs to the set of peripheral identification registers. It contains the RevAnd and Customer_Modified bit fields.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: minor fix indicator

Indicates minor errata fixes specific to the design, for example metal fixes after implementation. In most cases this field is 0x0. Arm® recommends that the component designers ensure that the bit field can be changed by a metal fix if required, for example by driving the bit field from registers that reset to zero.

0x0: no metal fix in the component

Bits 3:0 **CMOD[3:0]**: customer modification indicator

When the component is a reusable IP, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

0x0: no modification made

STM component ID0 register (STM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_0[7:0]**: bits [31:24] of component identification
Component identification register, that indicates that the identification registers are present.
0x0D: identification value

STM component ID1 register (STM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component class
Indicates the class of the component, for example, ROM table or CoreSight™ component.
0x9: indicates that the component is a CoreSight™ component

Bits 3:0 **PRMBL_1[3:0]**: bits [19:16] of component identification.
Component identification register, that indicates that the identification registers are present.
0x0: identification value

STM component ID2 register (STM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_2[7:0]**: bits [15:8] of component identification

Component identification register, that indicates that the identification registers are present.

0x05: identification value

STM component ID3 register (STM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_3[7:0]**: bits [7:0] of component identification

Component identification register, that indicates that the identification registers are present.

0xB1: identification value

System trace macrocell registers map

Table 587. STM registers map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD00	**STM_HEER**	HEE[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xD04 to 0xD1C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xD20	**STM_HETER**	HETE[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0xD24 to 0xD5C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xD60	**STM_HEBSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HEBS
	Reset value																																-
0xD64	**STM_HEMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN	TRIGCLEAR	TRIGSTATUS	TRIGCTL	Res.	ERRDETECT	COMPEN	EN
	Reset value																									0	-	-	0		-	-	0

Table 587. STM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD68	**STM_HEEXTMUXR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTMUX[7:0]							
	Reset value																									-	-	-	-	-	-	-	-
0xD6C to 0xDF0	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xDF4	**STM_HEMASTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0xDF8	**STM_HEFEAT1R**	Res.	HEEXTMUXSIZE[2:0]			Res.	Res.	Res.	Res.	NUMHE[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HECOMP[1:0]		HEMASTR	HEERR	Res.	HETER
	Reset value		0	1	1					0	0	1	0	0	0	0	0	0										1	1	0	1		1
0xDFC	**STM_HEIDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VENDSPEC[3:0]				CLASSREV[3:0]				CLASS[3:0]			
	Reset value																					0	0	0	0	0	0	0	1	0	0	0	1
0xE00	**STM_SPER**	SPE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04 to 0xE1C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xE20	**STM_SPTER**	SPTE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE24 to 0xE5C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xE60	**STM_SPSCR**	PORTSEL[11:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTCTL[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-																			0	0
0xE64	**STM_SPMSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
	Reset value										-	-	-	-	-	-	-	-															0

Table 587. STM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE68	**STM_SPOVERRIDER**	PORTSEL[16:0]																	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVERTS	OVERCTL[1:0]	
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-													0	0	0
0xE6C	**STM_SPMOVERRIDER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
	Reset value										-	-	-	-	-	-	-	-															0
0xE70	**STM_SPTRIGCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN_DIR	ATBTRIGEN_TE	TRIGCLEAR	TRIGSTATUS	TRIGCTL
	Reset value																												0	0	-	-	0
0xE74 to 0xE7C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xE80	**STM_TCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPEN	Res.	Res.	SYNCEN	TSEN	EN
	Reset value									0	-	-	-	-	-	-	-											0			1	0	0
0xE84	**STM_TSSTIMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCETS
	Reset value																																-
0xE88	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xE8C	**STM_TSFREQR**	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE90	**STM_SYNCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	COUNT[8:0]									Res.	Res.	Res.
	Reset value																				0	0	0	0	0	0	0	0	0	0			
0xE94	**STM_AUXCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QHWEVOVERRIDE	Res.	Res.	Res.	Res.	PRIORINVDIS	ASYNCPE	FIFOAF
	Reset value																									0					0	0	0
0xE98 to 0xE9C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Table 587. STM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xEA0	**STM_FEAT1R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOEN[1:0]		SYNCEN[1:0]		HWTEN[1:0]		TSPRESCALE[1:0]		TRIGCTL[1:0]		TRACEBUS[3:0]				SYNC[1:0]		FORCETS	TSFREQ	TS[1:0]		PROT[3:0]			
	Reset value									0	1	1	0	0	1	0	1	1	0	0	0	0	1	1	1	1	1	0	1	0	0	0	1
0xEA4	**STM_FEAT2R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPTYPE[1:0]		DSIZE[3:0]				Res.	SPTRTYPE[1:0]		PRIVMASK[1:0]		SPOVERRIDE	SPCOMP[1:0]		Res.	SPER	SPTER[1:0]	
	Reset value															0	1	0	0	0	1		1	0	0	1	1	1	1		0	1	0
0xEA8	**STM_FEAT3R**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMMAST[6:0]						
	Reset value																										1	1	1	1	1	1	1
0xEAC to 0xEE4	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xEE8	**STM_ITTRIGGER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASYNCOUT_W	TRIGOUTHETE_W	TRIGOUTSW_W	TRIGOUTSPTE_W
	Reset value																													-	-	-	-
0xEEC	**STM_ITATBDATA0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATDATAM63_W	ATDATAM55_W	ATDATAM47_W	ATDATAM39_W	ATDATAM31_W	ATDATAM23_W	ATDATAM15_W	ATDATAM7_W	ATDATAM0_W
	Reset value																								-	-	-	-	-	-	-	-	-
0xEF0	**STM_ITATBCTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFVALIDM_R	ATREADYM_R
	Reset value																															-	-
0xEF4	**STM_ITATBID**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATIDM_W[6:0]						
	Reset value																										-	-	-	-	-	-	-
0xEF8	**STM_ITATBCTR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBYTESM_W[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	AFREADYM_W	ATVALIDM_W
	Reset value																						-	-	-							-	-

Table 587. STM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xEFC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xF00	STM_ITCTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IME
	Reset value																																0
0xF04 to 0xF9C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFA0	STM_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET[3:0]			
	Reset value																													1	1	1	1
0xFA4	STM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR[3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFAC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFB0	STM_LAR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFB4	STM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
	Reset value																														0	1	1
0xFB8	STM_AUTHSTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									1	-	1	-	1	-	1	-
0xFBC	STM_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				Res.	ARCHID[14:0]														
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	1		0	0	0	1	0	1	0	0	1	1	0	0	0	1	1
0xFC0 to 0xFC4	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFC8	STM_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMSP[16:0]																
	Reset value																1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFCC	STM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	1	0	0	0	1	1
0xFD0	STM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
	Reset value																									0	0	0	0	0	1	0	0

Table 587. STM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	**STM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
	Reset value																									0	1	1	0	0	0	1	1
0xFE4	**STM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	**STM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	**STM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	STM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**STM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**STM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**STM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

64.10.9 Microprocessor debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers which control the power and clock behavior in debug mode. Specifically it allows the debugger, or debug software, to:

- maintain the clock and power to the processors when in low power modes (Sleep, Stop or Standby)
- maintain the clock and power to the system debug and trace components when in low power modes
- stop the clock to certain peripherals (CAN, SMBUS timeout, Watchdogs, Timers, RTC) when either processor is stopped in debug mode

Low power mode emulation in debug

When entering a low power mode (CSleep, Stop, LP-Stop, LPLV-Stop or Standby), the clock to the processor, the subsystem clocks or, in the case of Standby mode, the core power supply may be switched off. This prevents any debug accesses to the unclocked domains. To simplify debugging of applications which use these power saving features, the clock and power can be maintained or not according to the setting of the DBGSTBY, DBGSTOP and DBGSLEEP bits in the DBGMCU_CR register. These bits can be accessed directly by the debugger or by debug software, meaning that the application software does not need to be modified.

If the system enters one of the power saving modes, and the corresponding register bit is set, the system enters and exits the low power state as normal. However the clock and power remain active, allowing the debugger full access.

Note: If the trace port interface (TPIU) is used, loss of trace data can occur if the system goes into Standby mode and switches off the core power before the TPIU FIFO is empty. The software can check the state of the FIFO to avoid this. However if the DBGSTBY bit is set while using the trace port, the trace continues to flush during Standby mode and no specific software action is required.

Peripheral clock “freeze” in debug

When one or more processors are halted in debug state, by default the peripherals continue to operate. This means that the system state at the moment the breakpoint was taken is overwritten and can not be recovered. Furthermore, certain peripherals require software intervention on a regular basis, without which they may time out and generate a reset (watchdogs, for example).

By setting the corresponding bits in the DBGMCU Peripheral Freeze registers, certain peripheral clocks can be stopped automatically as soon as one or both processors enter debug state. The clocks remain stopped until the processor is restarted.

The following peripherals support this feature:

- All timers (TIMn, LPTIMn)
- All I2C peripherals (to prevent SMBus timeout)
- FDCAN
- All watchdog timers (WWDG, IWDGn)
- Real time clock (RTC)

There is a pair of freeze registers for each peripheral bus (APB), to control the peripherals on that bus. One register of each pair controls whether the peripheral is sensitive to the Cortex®-A7 debug state, the other to the Cortex®-M4. It is recommended to use the register that corresponds to the processor responsible for rearming the peripheral timeout.

The Cortex®-M4 debug state corresponds to a high level on the M4 core **HALTED** signal (**HALTED** remains asserted while the core is in debug). That of the Cortex®-A7is an OR of the **DBGACK** signal from each core (the core asserts **DBGACK** to indicate that the system has entered debug state). Hence if either CA7 core enters debug and the corresponding bit in the Peripheral Freeze register is set, then the peripheral clock will be frozen.

The IWDG1 (secure watchdog for the CA7) is an exception to the above. When secure debugging is disabled (**spiden** = 0), any CPU core which is in secure state will not stop in debug mode until it exits secure state. For security reasons, when **spiden** = 0, the IWDG1 timer will not stop unless DBGACK signal is active for both CPU cores, meaning both cores have stopped and hence both cores are in non-secure state. When secure debugging is enabled (**spiden** = 1), the behavior is determined by the WDFZCTL bit in the DBGMCU_CR register.

The IWDG1 behavior is summarized in truth *Table 588*.

Table 588. IWDG1 behavior in debug

Inputs					IWDG1 frozen?
DBGMCU_APB5FZ1 .IWDG1	spiden	DBGMCU_CR .WDFZCTL	DEBUGACK (CA7-0)	DEBUGACK (CA7-1)	
0	X	X	X	X	N
1	0	X	0	X	N
1	0	X	X	0	N
1	0	X	1	1	Y
1	1	0	0	0	N
1	1	0	1	X	Y
1	1	0	X	1	Y
1	1	1	0	X	N
1	1	1	X	0	N
1	1	1	1	1	Y

Microprocessor debug unit registers

The DBGMCU registers are not reset by a system reset **nreset**, only by a power on reset **vcore_rst**, by the debugger using CDBGRSTREQ/ACK, or by dbg_rstn which is set by writing to the bit DBGRST of RCC Debug Configuration register (RCC_DBGCFGR). They are accessible to the debugger via the debug APB bus at base address 0xE0081000. They are also accessible by both processors at base address 0x50081000

Note: The DBGMCU is not a standard CoreSight™ component and so does not appear in the system ROM table.

DBGMCU identity code register (DBGMCU_IDC)

Address offset: 0x000

Reset value: 0xXXXX X500

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: revision

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: device ID
0x500: STM32MP15xx

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	WDFZCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw				rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTBY	DBGSTOP	DBGSLEEP
													rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TRGOEN:** external trigger output enable

This bit controls the direction of the DBTRGIO bi-directional trigger pin.

0: input. DBTRGIO is connected to DBTRGI

1: output. DBTRGIO is connected to DBTRGO

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **WDFZCTL:** IWDG1 secure debug behavior

This bit determines the behavior of the IWDG1 in debug mode when **spiden** = 1. Refer to *Table 588: IWDG1 behavior in debug* for details.

0: OR. Freeze watchdog timer when either CA7 core halts in debug mode

1: AND. Freeze watchdog timer only when both CA7 cores halt in debug mode

Bits 23:3 Reserved, must be kept at reset value.

Bit 2 **DBGSTBY:** Standby mode debug enable

0: normal operation. All clocks are disabled and the V_{DDCORE} domain powered down automatically in Standby mode.

1: automatic clock stop/power down disabled. All active clocks and oscillators continue to run during Standby mode, and the V_{DDCORE} domain supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1 **DBGSTOP:** Stop mode debug enable

0: normal operation. In Stop or CStop mode, clocks may be stopped.

1: automatic clock stop disabled. All active clocks and oscillators continue to run during Stop, LP-Stop or CStop mode, allowing full debug capability. On exit from low-power mode, the clock settings are set to the normal low-power mode exit state.

Bit 0 **DBGSLEEP:** Sleep mode debug enable

0: normal operation. Processor clock is stopped automatically in CSleep mode.

1: automatic clock stop disabled. Processor clock continues to run, allowing full debug capability.

Note: this bit only affects the Cortex®-M4, the Cortex®-A7 debug clock continues to run even when the core clocks are stopped (PxSTOP mode).

DBGMCU APB4 peripheral freeze register MPU (DBGMCU_APB4FZ1)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2	Res.	Res.
													rw		

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **IWDG2**: IWDG2 stop in debug

0: normal operation. IWDG2 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. IWDG2 is frozen while either Cortex®-A7 core is in debug mode. No accesses are allowed on its APB interface.

Bits 1:0 Reserved, must be kept at reset value.

DBGMCU APB1 peripheral freeze register MPU (DBGMCU_APB1FZ1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C5	I2C3	I2C2	I2C1	Res.	Res.
										rw	rw	rw	rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	WWDG1	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C5**: I2C5 SMBUS timeout stop in debug

0: normal operation. I2C5 SMBUS timeout continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. I2C5 SMBUS timeout is frozen while either Cortex®-A7 core is in debug mode.

Bit 20 **I2C3**: I2C3 SMBUS timeout stop in debug

0: normal operation. I2C3 SMBUS timeout continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. I2C3 SMBUS timeout is frozen while either Cortex®-A7 core is in debug mode.

Bit 19 **I2C2**: I2C2 SMBUS timeout stop in debug

0: normal operation. I2C2 SMBUS timeout continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. I2C2 SMBUS timeout is frozen while either Cortex®-A7 core is in debug mode.

Bit 18 **I2C1**: I2C1 SMBUS timeout stop in debug

0: normal operation. I2C1 SMBUS timeout continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. I2C1 SMBUS timeout is frozen while either Cortex®-A7 core is in debug mode.

Bits 17:11 Reserved, must be kept at reset value.

Bit 10 **WWDG1**: WWDG1 stop in debug

0: normal operation. WWDG1 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. WWDG1 is frozen while either Cortex®-A7 core is in debug mode.

Bit 9 **LPTIM1**: LPTIM1 stop in debug

0: normal operation. LPTIM1 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. LPTIM1 is frozen while either Cortex®-A7 core is in debug mode.

Bit 8 **TIM14**: TIM14 stop in debug

0: normal operation. TIM14 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM14 is frozen while either Cortex®-A7 core is in debug mode.

Bit 7 **TIM13**: TIM13 stop in debug

0: normal operation. TIM13 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM13 is frozen while either Cortex®-A7 core is in debug mode.

Bit 6 **TIM12**: TIM12 stop in debug

0: normal operation. TIM12 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM12 is frozen while either Cortex®-A7 core is in debug mode.

Bit 5 **TIM7**: TIM7 stop in debug

0: normal operation. TIM7 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM7 is frozen while either Cortex®-A7 core is in debug mode.

Bit 4 **TIM6**: TIM6 stop in debug

0: normal operation. TIM6 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM6 is frozen while either Cortex®-A7 core is in debug mode.

Bit 3 **TIM5**: TIM5 stop in debug

0: normal operation. TIM5 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM5 is frozen while either Cortex®-A7 core is in debug mode.

Bit 2 **TIM4**: TIM4 stop in debug

0: normal operation. TIM4 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM4 is frozen while either Cortex®-A7 core is in debug mode.

Bit 1 **TIM3**: TIM3 stop in debug

0: normal operation. TIM3 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM3 is frozen while either Cortex®-A7 core is in debug mode.

Bit 0 **TIM2**: TIM2 stop in debug

0: Normal operation. TIM2 continues to operate while Cortex®-A7 is in debug mode.
1: Stop in debug. TIM2 is frozen while either Cortex®-A7 core is in debug mode.

DBGMCU APB1 peripheral freeze register MCU (DBGMCU_APB1FZ2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C5	I2C3	I2C2	I2C1	Res.	Res.
										rw	rw	rw	rw		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	WWDG1	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **I2C5**: I2C5 SMBUS timeout stop in debug
0: normal operation. I2C5 SMBUS timeout continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. I2C5 SMBUS timeout is frozen while Cortex®-M4 is in debug mode.

Bit 20 **I2C3**: I2C3 SMBUS timeout stop in debug
0: normal operation. I2C3 SMBUS timeout continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. I2C3 SMBUS timeout **i**s frozen while Cortex®-M4 is in debug mode.

Bit 19 **I2C2**: I2C2 SMBUS timeout stop in debug
0: normal operation. I2C2 SMBUS timeout continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. I2C2 SMBUS timeout **i**s frozen while Cortex®-M4 is in debug mode.

Bit 18 **I2C1**: I2C1 SMBUS timeout stop in debug
0: normal operation. I2C1 SMBUS timeout continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. I2C1 SMBUS timeout **i**s frozen while Cortex®-M4 is in debug mode.

Bits 17:11 Reserved, must be kept at reset value.

Bit 10 **WWDG1**: WWDG1 stop in debug
0: normal operation. WWDG1 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. WWDG1 is frozen while Cortex®-M4 is in debug mode.

Bit 9 **LPTIM1**: LPTIM1 stop in debug
0: normal operation. LPTIM1 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. LPTIM1 **i**s frozen while Cortex®-M4 is in debug mode.

Bit 8 **TIM14**: TIM14 stop in debug
0: normal operation. TIM14 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM14 **i**s frozen while Cortex®-M4 is in debug mode.

Bit 7 **TIM13**: TIM13 stop in debug
0: normal operation. TIM13 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM13 is frozen while Cortex®-M4 is in debug mode.

Bit 6 **TIM12**: TIM12 stop in debug
0: normal operation. TIM12 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM12 is frozen while Cortex®-M4 is in debug mode.

Bit 5 **TIM7**: TIM7 stop in debug
0: normal operation. TIM7 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM7 is frozen while Cortex®-M4 is in debug mode.

Bit 4 **TIM6**: TIM6 stop in debug
0: normal operation. TIM6 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM6 is frozen while Cortex®-M4 is in debug mode.

Bit 3 **TIM5**: TIM5 stop in debug
0: normal operation. TIM5 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM5 is frozen while Cortex®-M4 is in debug mode.

Bit 2 **TIM4**: TIM4 stop in debug
0: normal operation. TIM4 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM4 is frozen while Cortex®-M4 is in debug mode.

Bit 1 **TIM3**: TIM3 stop in debug
0: normal operation. TIM3 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM3 is frozen while Cortex®-M4 is in debug mode.

Bit 0 **TIM2**: TIM2 stop in debug
0: normal operation. TIM2 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM2 is frozen while Cortex®-M4 is in debug mode.

DBGMCU APB2 peripheral freeze register MPU (DBGMCU_APB2FZ1)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FDCAN	Res.	Res.	Res.	Res.	Res.	Res.	TIM17	TIM16	TIM15	Res.	Res.	Res.	Res.	TIM8	TIM1
							rw	rw	rw					rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FDCAN**: FDCAN stop in debug
0: normal operation. FDCAN continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. FDCAN is frozen while either Cortex®-A7 core is in debug mode.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **TIM17**: TIM17 stop in debug
0: normal operation. TIM17 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM17 is frozen while either Cortex®-A7 core is in debug mode.

Bit 7 **TIM16**: TIM16 stop in debug
0: normal operation. TIM16 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM16 is frozen while either Cortex®-A7 core is in debug mode.

Bit 6 **TIM15**: TIM15 stop in debug
0: normal operation. TIM15 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM15 is frozen while either Cortex®-A7 core is in debug mode.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **TIM8**: TIM8 stop in debug

0: normal operation. TIM8 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM8 is frozen while either Cortex®-A7 core is in debug mode.

Bit 0 **TIM1**: TIM1 stop in debug

0: normal operation. TIM1 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. TIM1 is frozen while either Cortex®-A7 core is in debug mode.

DBGMCU APB2 peripheral freeze register MCU (DBGMCU_APB2FZ2)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FDCAN	Res.	Res.	Res.	Res.	Res.	Res.	TIM17	TIM16	TIM15	Res.	Res.	Res.	Res.	TIM8	TIM1
							rw	rw	rw					rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FDCAN**: FDCAN stop in debug

0: normal operation. FDCAN continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. FDCAN is frozen while Cortex®-M4 is in debug mode.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **TIM17**: TIM17 stop in debug

0: normal operation. TIM17 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM17 is frozen while Cortex®-M4 is in debug mode.

Bit 7 **TIM16**: TIM16 stop in debug

0: normal operation. TIM16 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM16 is frozen while Cortex®-M4 is in debug mode.

Bit 6 **TIM15**: TIM15 stop in debug

0: normal operation. TIM15 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM15 is frozen while Cortex®-M4 is in debug mode.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **TIM8**: TIM8 stop in debug

0: normal operation. TIM8 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM8 is frozen while Cortex®-M4 is in debug mode.

Bit 0 **TIM1**: TIM1 stop in debug

0: normal operation. TIM1 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. TIM1 is frozen while Cortex®-M4 is in debug mode.

DBGMCU APB3 peripheral freeze register MPU (DBGMCU_APB3FZ1)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.
											rw	rw	rw	rw	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LPTIM5**: LPTIM5 stop in debug

0: normal operation. LPTIM5 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. LPTIM5 is frozen while either Cortex®-A7 core is in debug mode.

Bit 3 **LPTIM4**: LPTIM4 stop in debug

0: normal operation. LPTIM4 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. LPTIM4 is frozen while either Cortex®-A7 core is in debug mode.

Bit 2 **LPTIM3**: LPTIM3 stop in debug

0: normal operation. LPTIM3 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. LPTIM3 is frozen while either Cortex®-A7 core is in debug mode.

Bit 1 **LPTIM2**: LPTIM2 stop in debug

0: normal operation. LPTIM2 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. LPTIM2 is frozen while either Cortex®-A7 core is in debug mode.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register MCU (DBGMCU_APB3FZ2)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.
											rw	rw	rw	rw	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **LPTIM5**: LPTIM5 stop in debug

0: normal operation. LPTIM5 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. LPTIM5 is frozen while Cortex®-M4 is in debug mode.

Bit 3 **LPTIM4**: LPTIM4 stop in debug

0: normal operation. LPTIM4 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. LPTIM4 is frozen while Cortex®-M4 is in debug mode.

Bit 2 **LPTIM3**: LPTIM3 stop in debug
0: normal operation. LPTIM3 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. LPTIM3 is frozen while Cortex®-M4 is in debug mode.

Bit 1 **LPTIM2**: LPTIM2 stop in debug
0: normal operation. LPTIM2 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. LPTIM2 is frozen while Cortex®-M4 is in debug mode.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB5 peripheral freeze register MPU (DBGMCU_APB5FZ1)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	I2C6	Res.	Res.	Res.	Res.	RTC	IWDG1	I2C4	Res.	Res.
						rw					rw	rw	rw		

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **I2C6**: I2C6 stop in debug
0: normal operation. I2C6 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. I2C6 is frozen while either Cortex®-A7 core is in debug mode.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **RTC**: RTC stop in debug
0: normal operation. RTC continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. RTC is frozen while either Cortex®-A7 core is in debug mode.

Bit 3 **IWDG1**: independent watchdog 1 stop in debug
0: normal operation. Watchdog continues to count while Cortex®-A7 is in debug mode.
1: stop in debug. IWGD1 is frozen while either Cortex®-A7 core is in debug mode (depending on the state of WDGFZCTL bit in DBGMCU_CR). No accesses are allowed on its APB interface.

Bit 2 **I2C4**: I2C4 stop in debug
0: normal operation. I2C4 continues to operate while Cortex®-A7 is in debug mode.
1: stop in debug. I2C4 is frozen while either Cortex®-A7 core is in debug mode.

Bits 1:0 Reserved, must be kept at reset value.

DBGMCU APB5 peripheral freeze register MCU (DBGMCU_APB5FZ2)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	I2C6	Res.	Res.	Res.	Res.	RTC	Res.	I2C4	Res.	Res.
						rw					rw		rw		

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **I2C6**: I2C6 stop in debug

0: normal operation. I2C6 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. I2C6 is frozen while Cortex®-M4 is in debug mode.

Bits 8:5 Reserved, must be kept at reset value.

Bit 4 **RTC**: RTC stop in debug

0: normal operation. RTC continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. RTC is frozen while Cortex®-M4 is in debug mode.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **I2C4**: I2C4 stop in debug

0: normal operation. I2C4 continues to operate while Cortex®-M4 is in debug mode.
1: stop in debug. I2C4 is frozen while Cortex®-M4 is in debug mode.

Bits 1:0 Reserved, must be kept at reset value.

Microprocessor controller debug unit registers map

Table 589. DBGMCU registers map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DBGMCU_IDC**	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					0	1	0	1	0	0	0	0	0	0	0	0
0x004	**DBGMCU_CR**	Res.	Res.	Res.	TRGOEN	Res.	Res.	Res.	WDFZCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGSTBY	DBGSTOP	DBGSLEEP
	Reset value				0				0																						0	0	0
0x008 to 0x028	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x02C	**DBGMCU_ APB4FZ1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IWDG2	Res.	Res.
	Reset value																														0		
0x030	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x034	**DBGMCU_ APB1FZ1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C5	I2C3	I2C2	I2C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
	Reset value											0	0	0	0								0	0	0	0	0	0	0	0	0	0	0

Table 589. DBGMCU registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	DBGMCU_ APB1FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C5	I2C3	I2C2	I2C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WWDG1	LPTIM1	TIM14	TIM13	TIM12	TIM7	TIM6	TIM5	TIM4	TIM3	TIM2
	Reset value											0	0	0	0								0	0	0	0	0	0	0	0	0	0	0
0x03C	DBGMCU_ APB2FZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN	Res.	Res.	Res.	Res.	Res.	Res.	TIM17	TIM16	TIM15	Res.	Res.	Res.	Res.	TIM8	TIM1
	Reset value																	0							0	0	0					0	0
0x040	DBGMCU_ APB2FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN	Res.	Res.	Res.	Res.	Res.	Res.	TIM17	TIM16	TIM15	Res.	Res.	Res.	Res.	TIM8	TIM1
	Reset value																	0							0	0	0					0	0
0x044	DBGMCU_ APB3FZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.
	Reset value																												0	0	0	0	
0x048	DBGMCU_ APB3FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPTIM5	LPTIM4	LPTIM3	LPTIM2	Res.
	Reset value																												0	0	0	0	
0x04C	DBGMCU_ APB5FZ1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C6	Res.	Res.	Res.	Res.	RTC	IWDG1	I2C4	Res.	Res.
	Reset value																							0					0	0	0		
0x050	DBGMCU_ APB5FZ2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	I2C6	Res.	Res.	Res.	Res.	RTC	Res.	I2C4	Res.	Res.
	Reset value																							0					0		0		
0x054- 0xFFC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

64.11 Cortex®-A7 debug features

The Cortex®-A7 subsystem integrates the following CoreSight™ components, one of each per core:

- Debug unit (DBG)
- Performance Monitoring Unit (PMU)
- Embedded Trace Macrocell (ETM)
- Cross Trigger Interface (CTI)

These components are accessible by the debugger via the APB-AP and its associated debug APB bus. They are also accessible by both Cortex®-A7 and Cortex®-M4 via the system interconnect.

64.11.1 Cortex®-A7 debug unit

The debug unit enables the debugger to access the debug features buit into the Cortex®-A7 MPCore. The debug features allow an external debugger to:

- Stop program execution
- Examine and alter process and coprocessor state
- Examine and alter memory and input/output peripheral state
- Restart the processor

For detailed information on the debug unit, refer to the *Cortex®-A7 MPCore Technical Reference Manual [9]*.

Debug unit registers

The debug unit comprises a number of registers accessible to the debugger via the debug APB at address range 0xE00D 0000 to 0xE00D 0FFC for CA7-CPU1, and 0xE00D 2000 to 0xE00D 2FFC for CA7-CPU2. The same registers are accessible to software at address range 0x500D 0000 to 0x500D 0FFC for CA7-CPU1, and 0x500D 2000 to 0x500D 2FFC for CA7-CPU2.

Each Cortex®-A7 core can access certain of its own debug unit registers via the CP14 (baseline and extended) registers, using MCR, MRC, MCCR and MRRC instructions.

DBG identification register (DBG_DIDR)

Address offset: 0x000

Reset value: 0x3515 F005

Specifies which version of debug architecture is implemented and what features.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPS[3:0]				BRPS[3:0]				CTX_CMPS[3:0]				VERSION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEVIDIMP	NSUHDIMP	PCSRIMP	SEIMP	Res.	Res.	Res.	Res.	VARIANT[3:0]				REVISION[3:0]			
r	r	r	r					r	r	r	r	r	r	r	r

Bits 31:28 **WRPS[3:0]**: Watchpoint Registers Pairs.
Indicates the number of watchpoint register pairs implemented.
0x3: The processor implements 4 WRPs

Bits 27:24 **BRPS[3:0]**: Breakpoint Registers Pairs.
Indicates the number of breakpoint register pairs implemented.
0x5: The processor implements 6 BRPs

Bits 23:20 **CTX_CMPS[3:0]**: Context ID comparators.
Indicates the number of BRPs that can be used for Context ID comparison.
0x1: The processor implements 2 breakpoints with context ID comparison

Bits 19:16 **VERSION[3:0]**: Debug Architecture version.
0x5: Arm®v7.1 debug architecture

Bit 15 **DEVIDIMP**: Debug Device ID register (DBGDEVID) implemented.
1: Implemented

Bit 14 **NSUHDIMP**: Secure User Halting Debug implemented.
1: Not implemented

Bit 13 **PCSRIMP**: Program Counter Sampling Register (DBGPCSR) implemented as register 33.
1: Implemented

Bit 12 **SEIMP**: Security Extensions implemented.
1: Implemented

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **VARIANT[3:0]**: Processor Variant number.
This number is incremented on functional changes. This field matches bits [23:20] of the CP15 Main ID register.
0x0: Variant = 0

Bits 3:0 **REVISION[3:0]**: Processor Revision number.
This number is incremented on bug fixes. This field matches bits [3:0] of the CP15 Main ID register.
0x5: Revision = 5

DBG watchpoint fault address register (DBG_WFAR)

Address offset: 0x018

Reset value: 0x0000 0000

Returns information about the address of the instruction that accessed a watchpointed address. Not used in this version of debug architecture.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ADDRESS[31:0]**: Instruction address.
This register is unused.

DBG vector catch register (DBG_VCR)

Address offset: 0x01C

Reset value: 0x0000 0000

Controls Vector catch debug events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSF	NSI	Res.	NSD	NSP	NSC	NSU	Res.	NSHF	NSHI	NSHE	NSHD	NSHP	NSHC	NSHU	Res.
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MF	MI	Res.	MD	MP	MS	Res.	Res.	SF	SI	Res.	SD	SP	SS	SU	R
rw	rw		rw	rw	rw			rw	rw		rw	rw	rw	rw	rw

Bit 31 **NSF**: FIQ interrupt exception non-secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 30 **NSI**: IRQ interrupt exception non-secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **NSD**: Data Abort exception non-secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 27 **NSP**: Prefetch Abort exception non-secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 26 **NSC**: Supervisor Call exception non-secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 25 **NSU**: Undefined Instruction exception non-secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 24 Reserved, must be kept at reset value.

Bit 23 **NSHF**: FIQ interrupt exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 22 **NSHI**: IRQ interrupt exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 21 **NSHE**: Hyp Trap or Hyp mode entry exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 20 **NSHD**: Data Abort exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 19 **NSHP**: Prefetch Abort exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 18 **NSHC**: Hypervisor Call exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 17 **NSHU**: Undefined Instruction exception non-secure Hyp Vector catch enable.
0: Disabled
1: Enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **MF**: FIQ interrupt exception secure monitor mode Vector catch enable.
0: Disabled
1: Enabled

Bit 14 **MI**: IRQ interrupt exception secure monitor mode Vector catch enable.
0: Disabled
1: Enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **MD**: Data Abort exception secure monitor mode Vector catch enable.
0: Disabled
1: Enabled

Bit 11 **MP**: Prefetch Abort exception secure monitor mode Vector catch enable.
0: Disabled
1: Enabled

Bit 10 **MS**: Secure Monitor Call exception Vector catch enable.
0: Disabled
1: Enabled

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **SF**: FIQ interrupt exception secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 6 **SI**: IRQ interrupt exception secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **SD**: Data Abort exception secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 3 **SP**: Prefetch Abort exception secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 2 **SS**: Supervisor Call exception secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 1 **SU**: Undefined Instruction exception secure local Vector catch enable.
0: Disabled
1: Enabled

Bit 0 **R**: Reset Vector catch enable.
0: Disabled
1: Enabled

DBG event catch register (DBG_ECR)

Address offset: 0x024

Reset value: 0x0000 0000

Configures the debug logic to generate the OS Unlock catch debug event when the OS Lock is cleared.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSUCE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **OSUCE**: OS Unlock Catch enable.
An OS Unlock catch debug event is generated when the OS Lock is cleared by writing to the DBGOSLAR.
0: Disabled
1: Enabled

DBG host to target data transfer register (DBG_DTRRX)

Address offset: 0x080

Reset value: 0x0000 0000

Transfers data from an external host to the Arm® processor. For example it is used by a debugger transferring commands and data to a debug target. It is a component of the Debug Communication Channel (DCC).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Host to Target Data.
One word of data for transfer from the debug host to the debug target.

DBG instruction transfer register (DBG_ITR)

Address offset: 0x084

Reset value: 0x0000 0000

When the processor is in Debug state, transfers an Arm® instruction to the processor for execution.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSTRUCTION[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRUCTION[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **INSTRUCTION[31:0]**: Instruction.
32-bit encoded Arm® instruction to execute on the processor

DBG status and control register (DBG_DSCR)

Address offset: 0x088

Reset value: 0x0000 0000

The main control register for debug.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXFULL	TXFULL	Res.	RXFULL_L	TXFULL_L	PIPEADV	INSTRCOMPL_L	Res.	Res.	EXTDCCMODE[1:0]		ADADISCARD	NS	SPNIDDIS	SPIDDIS
	rw	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDBGEN	HDBGEN	ITREN	UDCCDIS	INTDIS	DBGACK	FS	UND_L	ADABORT_L	SDABORT_L	MOE[3:0]				RESTARTED	HALTED
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **RXFULL**: DBGDTRRX register full.
0: Register empty
1: Register full

Bit 29 **TXFULL**: DBGDTRTX register full.
0: Register empty
1: Register full

Bit 28 Reserved, must be kept at reset value.

Bit 27 **RXFULL_L**: Latched RXfull.

This controls the behavior of the processor on writes to DBGDTRRX in non-blocking mode (DBGDSCR.EXTDCCMODE = 0x0)

0: Writes to DBGDTRRX execute and set the RXFULL and RXFULL_L bits to 1

1: Writes to DBGDTRRX are ignored

Bit 26 **TXFULL_L**: Latched TXfull.

This controls the behavior of the processor on reads to DBGDTRTX in non-blocking mode (DBGDSCR.EXTDCCMODE = 0x0)

0: Reads to DBGDTRTX return an undefined value

1: Reads to DBGDTRTX return the register contents and reset the TXFULL and TXFULL_L bits to 0

Bit 25 **PIPEADV**: Sticky Pipeline Advance (Read only).

This bit is set to 1 whenever the processor pipeline advances by retiring one or more instructions. It is cleared to 0 only by a write to DBGDRCR.CSPA. This bit enables a debugger to detect that the processor is idle. In some situations this might indicate that the processor is deadlocked.

Bit 24 **INSTRCOMPL_L**: Latched Instruction Complete (Read Only).

This is a copy of the internal InstrCompl flag, taken on each read of DBGDSCR InstrCompl signals whether the processor has completed execution of an instruction issued through DBGITR. InstrCompl is not visible directly in any register. On a read of DBGDSCR when the processor is in Debug state, InstrCompl_l always returns the current value of InstrCompl. The meanings of the values of InstrCompl_l are:

0: An instruction previously issued through the DBGITR has not completed its changes to the architectural state of the processor.

1: All instructions previously issued through the DBGITR have completed their changes to the architectural state of the processor.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **EXTDCCMODE[1:0]**: External DCC Access Mode.

This field controls the access mode for the DCC registers and the DBG_ITR.

0x0: Non-blocking mode. Accesses always complete without stalling the bus.

0x1: Stall mode. Accesses stall the bus until they can complete successfully.

0x2: Fast mode. Allows an instruction to be latched in the DBG_ITR and executed repeatedly by successive read accesses to DBG_DTRTX, or write accesses to DBG_DTRRX

0x3: Reserved

Bit 19 **ADADISCARD**: Asynchronous Aborts Discarded.

0: Asynchronous aborts handled normally

1: On an asynchronous abort to which this bit applies, the processor sets the Sticky Asynchronous Abort bit, ADABORT_L, to 1 but otherwise discards the abort.

Bit 18 **NS**: Non-Secure state status (Read Only).

0: Processor is in Secure state

1: Processor is in Non-Secure state

Bit 17 **SPNIDDIS**: Secure PL1 Non-Invasive Debug Disabled (Read Only).

This bit shows if non-invasive debug is permitted in Secure PL1 modes.

0: Non-invasive debug permitted in Secure PL1 modes

1: Non-invasive debug not permitted in Secure PL1 modes

Bit 16 **SPIDDIS**: Secure PL1 Invasive Debug Disabled (Read Only).
This bit shows if invasive debug is permitted in Secure PL1 modes.
0: Invasive debug permitted in Secure PL1 modes
1: Invasive debug not permitted in Secure PL1 modes

Bit 15 **MDBGEN**: Monitor Debug mode enable.
0: Monitor debug mode disabled
1: Monitor debug mode enabled

Bit 14 **HDBGEN**: Halting Debug mode enable.
0: Halting debug mode disabled
1: Halting debug mode enabled

Bit 13 **ITREN**: Enable Arm® instruction execution.
This bit enables the execution of Arm® instructions through the DBGITR. This bit should always be written low when the processor is not in debug state.
0: ITR mechanism disabled
1: ITR mechanism enabled

Bit 12 **UDCCDIS**: User mode access to Debug Communications Channel (DCC) disable.
0: DCC access in User mode enabled
1: User mode acces to any DCC register via CP14 will generate an undefined instruction exception

Bit 11 **INTDIS**: Interrupt disable.
0: FIQ and IRQ enabled
1: FIG and IRQ masked

Bit 10 **DBGACK**: Force Debug Acknowledge.
This allows the debugger to simulate debug mode entry via the cross-trigger while the core remains in non-debug mode.
0: Normal operation
1: DBGACK asserted

Bit 9 **FS**: Fault Status (Read Only).
This bit is updated when a data abort exception is generated in debug state. It indicates where the fault information is held.
0: Fault status information may be held in PL1 fault reporting registers (DFSR and DFAR) or PL2 fault syndrome registers (HSR, HDFAR) according to current state, mode and the HCR.TGE register value
1: Fault status information was written to the PL2 fault syndrome registers

Bit 8 **UND_L**: Sticky Undefined Instruction (Read Only).
This bit is set to 1 by any Undefined Instruction exceptions generated by instructions issued to the processor while in Debug state.
0: No Undefined Instruction exception has been generated since the last time this bit was cleared
1: An Undefined Instruction exception has been generated since the last time this bit was cleared

Bit 7 **ADABORT_L**: Sticky Asynchronous Abort (Read Only).
This bit is set to 1 by any asynchronous abort that occurs while in Debug state, if ADADISCARD bit is set to 1.
0: No Asynchronous Abort has been generated since the last time this bit was cleared
1: An Asynchronous Abort has been generated since the last time this bit was cleared

Bit 6 **SDABORT_L**: Sticky Synchronous Data Abort (Read Only).
This bit is set to 1 by any synchronous data abort exception generated while in Debug state.
0: No synchronous Data Abort has been generated since the last time this bit was cleared
1: A synchronous Data Abort has been generated since the last time this bit was cleared

Bits 5:2 **MOE[3:0]**: Method of Debug Entry (Read Only).
Indicates the debug event which caused the processor to enter into debug mode.
0x0: Halt request
0x1: Breakpoint
0x2: Asynchronous watchpoint
0x3: BKPT instruction
0x4: External debug request
0x5: Vector catch
0x6 to 0x7: Reserved
0x8: OS unlock catch
0x9: Reserved
0xA: Synchronous watchpoint
0xB to 0xF: Reserved

Bit 1 **RESTARTED**: Processor Restarted (Read Only).
After making a restart request, the debugger can poll this bit until it is set to 1. At that point it knows that the restart request has taken effect and the processor has exited Debug state.
0: The processor is exiting Debug state. This bit only reads as 0 between receiving a restart request, and restarting Non-debug state operation
1: The processor has exited Debug state. This bit remains set to 1 if the processor re-enters Debug state.

Bit 0 **HALTED**: Processor Halted (Read Only).
After programming a debug event, the external debugger can poll this bit until it is set to 1. At that point it knows that the processor has entered Debug state.
0: The processor is in Non-debug state
1: The processor is in Debug state

DBG target to host data transfer register (DBG_DTRTX)

Address offset: 0x08C

Reset value: 0x0000 0000

Transfers data from the Arm® processor to an external host. For example it is used by a debug target to transfer commands and data to a debugger. It is a component of the Debug Communication Channel (DCC).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Target to Host Data.
One word of data for transfer from the debug target to the debug host.

DBG run control register (DBG_DRCR)

Address offset: 0x090

Reset value: 0x0000 0000

Software uses this register to request the processor to enter or exit Debug state, and to clear the sticky exception bits in the DBG_DSCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBRRQ	CSPA	CSE	RRQ	HRQ
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CBRRQ**: Cancel Bus Requests.
- 0: No action
- 1: Request cancel of pending accesses

Bit 3 **CSPA**: Clear Sticky Pipeline Advance bit.
- 0: No action
- 1: Reset the DBG_DSCR.PIPEADV bit to 0

Bit 2 **CSE**: Clear Sticky Exception bits..
- 0: No action
- 1: Resets the DBG_DSCR[8:6] bits to 0

Bit 1 **RRQ**: Restart Request.
This request is held until the processor exits Debug state. After the request has been made, the debugger can poll the DBG_DSCR.RESTARTED bit until it reads as 1.
- 0: No action
- 1: Request exit from debug mode.

Bit 0 **HRQ**: Halt Request.
This request is held until the processor enters Debug state. After the request has been made, the debugger can poll the DBG_DSCR.HALTED bit until it reads as 1.
- 0: No action
- 1: Request entry from debug mode.

DBG external auxiliary control register (DBG_EACR)

Address offset: 0x094

Reset value: 0x0000 0000

Provides implementation-defined configuration and control options.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDRSTAT	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CDRSTAT**: Core Debug Reset Status (Read Only).
Reflects the current reset state of the debug logic in the CPU power domain.
0: Debug logic in CPU power domain is not in reset state
1: Debug logic in CPU power domain is in reset state

Bits 2:0 Reserved, must be kept at reset value.

DBG program counter sampling register (DBG_PCSR)

Address offset: 0x0A0

Reset value: 0x0000 0000

This register provides implementation-defined configuration and control options.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCS[15:1]															T
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:1 **PCS[31:1]**: Program Counter Sample value.
The sampled value of bits[31:1] of the PC. The sampled value is either the virtual address of an instruction, or the virtual address of an instruction address plus an offset that depends on the processor instruction set state. DBG_DEVID1.PCSROFFSET indicates whether an offset is applied to the sampled addresses.

Bit 0 **T**: Thumb instruction.
This bit indicates whether the sampled address is an Arm® instruction, or a Thumb or ThumbEE instruction:
0: Arm® instruction
1: Thumb or ThumbEE instruction

DBG context ID sampling register (DBG_CIDSR)

Address offset: 0x0A4

Reset value: 0x0000 0000

Samples the CONTEXTIDR whenever the DBG_PCSR samples the program counter. This enables a debugger to associate a program counter sample with the process running on the processor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONTEXTIDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTEXTIDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CONTEXTIDR[31:0]**: Context ID Register sample value.

The value of the Context ID Register, CONTEXTIDR, associated with the last PC sample read from DBG_PCSR.

DBG virtualisation id sampling register (DBG_VIDSR)

Address offset: 0x0A8

Reset value: 0x0000 0000

Samples the VMID, Hyp mode status, and NS state whenever the DBGPCSR samples the program counter. This enables a debugger to associate a program counter sample with the virtual machine running on the processor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NS	H	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **NS**: Sampled NS state.

Indicates the Secure or Non-secure state associated with the last PC sample read from DBGPCSR.

0: Secure state

1: Non-secure state

Bit 30 **H**: Sampled Hyp mode. Indicates whether the last PC sample read from DBG_PCSR was associated with Hyp mode.

0: Not Hyp mode

1: Hyp mode

Bits 29:8 Reserved, must be kept at reset value.

Bits 7:0 **VMID[7:0]**: VMID sample.

The value of the VMID field from the VTTBR, associated with the last PC sample read from DBG_PCSR.

DBG breakpoint value x register (DBG_BVRx)

Address offset: 0x100 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Holds a value for use in breakpoint matching, either an instruction address or a Context ID. Each DBG_BVR is associated with a DBG_BCR to form a Breakpoint Register Pair (BRP) ie. DBG_BVR[n] is associated with DBG_BCR[n] to form breakpoint n.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESSCONTEXTID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESSCONTEXTID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESSCONTEXTID[31:0]**: Instruction Address or Context ID.
If the register is being used for address comparison, this field contains the instruction address to be matched (bits 31:2; bits 1:0 must be set to zero). If the register is being used for context ID comparison, this field contains the Context ID to be matched.

DBG breakpoint control x register (DBG_BCRx)

Address offset: 0x140 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Holds control information for breakpointing. Each DBG_BCR is associated with a DBG_BVR to form a Breakpoint Register Pair (BRP) ie. DBG_BCR[n] is associated with DBG_BVR[n] to form breakpoint n.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSC[1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC[1:0]		E
rw	rw	rw					rw	rw	rw	rw			rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **BT[3:0]**: Breakpoint Type.
Determines the type of comparison made by the breakpoint.
0x0: Unlinked instruction address match
0x1: Linked instruction address match
0x2: Unlinked Context ID match
0x3: Linked Context ID match
0x4: Unlinked instruction address mismatch
0x5: Linked instruction address mismatch
0x6-0x7: Reserved
0x8: Unlinked VMID match
0x9: Linked VMID match
0xA: Unlinked VMID match and Context ID match
0xB: Linked VMID match and Context ID match
0xC-0xF: Reserved

Bits 19:16 **LBN[3:0]**: Linked Breakpoint Number.
If this breakpoint is programmed for Linked instruction address match or mismatch then this field must be programmed with the number of the breakpoint that holds the Context match to be used in the combined instruction address and Context comparison. Otherwise, this field must be programmed to 0.

Bits 15:14 **SSC[1:0]**: Security State Control.
This field enables the breakpoint to be conditional on the security state of the processor. This field must be programmed to 0 if DBG_BCR.BT is programmed for Linked Context ID match.
0: Secure and Non-secure state
1: Non-secure state only
2: Secure state only
3: Hyp mode only (with HMC=1 and PMC = 0)

Bit 13 **HMC**: Hyp Mode Control.
This field controls whether the breakpoint matches in Hyp mode.
This field must be programmed to 0 if DBG_BCR.BT is programmed for Linked Context ID match.
0: Breakpoint does not match in Hyp mode
1: Breakpoint can match in Hyp mode

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:5 **BAS[3:0]**: Byte Address Select.
This field enables match or mismatch comparisons on only certain bytes of the word address held in the DBG_BVR. The operation of this field depends also on:
The breakpoint type field being programmed for instruction address match or mismatch.
The instruction set state of the processor, indicated by the CPSR.J and CPSR.T bits.
This field must be programmed to 0xF if the DBG_BVR.BT is programmed for Linked or Unlinked Context ID match.

Bits 4:3 Reserved, must be kept at reset value.

Bits 2:1 **PMC[1:0]**: Privileged Mode Control.
This field enables the breakpoint to be conditional on the mode of the processor. This field must be programmed to 3 if DBG_BCR.BT is programmed for Linked Context ID match.
0: PL0, Supervisor and System modes only
1: PL1 and PL2 modes only
2: PL0 mode only
3: All modes

Bit 0 **E**: Breakpoint Enable.
0: Breakpoint disabled
1: Breakpoint enabled

DBG watchpoint value x register (DBG_WVRx)

Address offset: 0x180 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Holds a value for use in Watchpoint matching, The address used must be the virtual address of the data. Each DBG_WVR is associated with a DBG_WCR to form a Breakpoint Register Pair (BRP) ie. DBG_WVR[n] is associated with DBG_WCR[n] to form breakpoint n.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRESS[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:2 **ADDRESS[31:2]**: Data Address.
This field indicates bits[31:2] of the address value for comparison.

Bits 1:0 Reserved, must be kept at reset value.

DBG watchpoint control xregister (DBG_WCRx)

Address offset: 0x1C0 + 0x4 * x (x=0 to 3)

Reset value: 0x0000 0000

Holds control information for watchpoint. Each DBG_WCR is associated with a DBG_WVR to form a Breakpoint Register Pair (BRP) ie. DBG_WCR[n] is associated with DBG_WVR[n] to form breakpoint n.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	MASK[4:0]					Res.	Res.	Res.	WT	LBN[3:0]			
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSC[1:0]		HMC	BAS[7:0]								LSC[1:0]		PAC[1:0]		E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **MASK[4:0]**: Address Mask.
This field can be used to set a watchpoint on a range of addresses by masking lower order address bits out of the watchpoint comparison. The value of this field is the number of low order bits of the address that are masked off, except that values of 1 and 2 are reserved.
0x0: No mask
0x1: Reserved
0x2: Reserved
0x3: Three bits masked (data address mask = 0x00000007)
0x4: Four bits masked (data address mask = 0x0000000F)
0x5: Five bits masked (data address mask = 0x0000001F)
..
0x1F: 31 bits masked (data address mask = 0x7FFFFFFF)

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **WT**: Watchpoint Type.
This bit is set to 1 to link the watchpoint to a breakpoint to create a linked watchpoint that requires both data address matching and Context ID matching. The linked breakpoint number field indicates the breakpoint to which this watchpoint is linked.
0: Linking disabled.
1: Linking enabled.

Bits 19:16 **LBN[3:0]**: Linked Breakpoint Number.
If this watchpoint is programmed with watchpoint type set to linkink enabled, then this field must be programmed with the number of the breakpoint that defines the Context match to be combined with data address comparison. Otherwise, this field must be programmed to 0.

Bits 15:14 **SSC[1:0]**: Security State Control.
This field enables the watchpoint to be conditional on the security state of the processor.
0: Secure and Non-secure state
1: Non-secure state only
2: Secure state only
3: Hyp mode only (with HMC=1 and PAC = 0)

Bit 13 **HMC**: Hyp Mode Control.
This field controls whether the watchpoint matches in Hyp mode.
0: Watchpoint does not match in Hyp mode
1: Watchpoint can match in Hyp mode

Bits 12:5 **BAS[7:0]**: Byte Address Select.
A DBG_WVR is programmed with a word-aligned address. This field enables the watchpoint to hit only if certain bytes of the addressed word are accessed. The watchpoint hits if an access hits any byte being watched, even if:
The access size is larger than the size of the region being watched.
The access is unaligned, and the base address of the access is not in the same word of memory as the address in the DBG_WVR.
Watched bytes should be contiguous within a word ie. 11100001 should not be programmed.
Example values for BAS:
00000001: Watch byte 0
00000011: Watch bytes 1 and 0
00000010: Watch byte 1
00000111: Watch bytes 2, 1 and 0
11110000: Watch bytes 7,6,5 and 4
etc.

Bits 4:3 **LSC[1:0]**: Load/store access control.
This field enables watchpoint matching conditional on the type of access being made.
0: Reserved
1: Match on any Load, Load-Exclusive or Swap
2: Match on any Store, Store-Exclusive, or Swap
3: Match on either type of access

Bits 2:1 **PAC[1:0]**: Privileged Access Control.
This field enables watchpoint matching conditional on the mode of the processor.
0: PL0, Supervisor and System modes only
1: PL1 and PL2 modes only
2: PL0 mode only
3: All modes

Bit 0 **E**: Watchpoint Enable.
0: Watchpoint disabled
1: Watchpoint enabled

DBG breakpoint extended value x register (DBG_BXVRx)

Address offset: 0x250 + 0x4 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Holds a value for use in VMID matching for BRP that supports Context ID comparison. Each DBG_BXVR is associated with a DBG_BCR/DBG_BVR Breakpoint Register Pair (BRP) ie. DBG_BXVR[n] is associated with DBG_BCR[n] and DBG_BVR[n] to form breakpoint n.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **VMID[7:0]**: Virtual Machine ID.
Used to compare with the Virtual Machine ID (VMID) field, held in the Virtualization Translation Table Base Register (VTTBR).

DBG OS lock access register (DBG_OSLAR)

Address offset: 0x300

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSLOCKACCESS[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSLOCKACCESS[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **OSLOCKACCESS[31:0]**: DBG registers lock access
0xC5ACCE55: Disable access
Other values: Enable access

DBG OS lock status register (DBG_OSLSR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSLM [1]	NTT	OSLK	OSLM [0]
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 2 **NTT**: Need 32-bit.
0: 32-bit access is needed to write the key to the OS Lock Access Register.

Bit 1 **OSLK**: OS lock status
0: LOCK not set.
1: LOCK set. Access to DBG registers disabled

Bits 3, 0 **OSLM[1:0]**: OS lock model
This field identifies the form of OS Save and Restore mechanism implemented:
10: The processor implements the OS Lock Model but does not implement DBGOSSRR
Others: Reserved

DBG device power-down and reset control register (DBG_PRCR)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CORE PURQ	HCWR	CWRR	CORE NPDR Q
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **COREPURQ**: Power-up request
This bit enables a debugger to request that the power controller powers up the processor, enabling access to debug registers in the processor power domain:
0: DBGPWRUPREQ is low
1: DBGPWRUPREQ is high

Bit 2 **HCWR**: Hold reset
0: Do not hold the non-debug logic reset on power-up or warm reset.
1: Hold the non-debug logic of the processor in reset on power-up or warm reset. The processor is held in this state until this bit is set to 0

Bit 1 **CWRR**: Reset request
0: No action
1: Request internal reset using the memory mapped interface or CP14

Bit 0 **CORENPDRQ**: Hold power-up request
0: DBGNOPWRDWN is low
1: DBGNOPWRDWN is high

DBG device power-down and reset status register (DBG_PRSR)

Address offset: 0x314

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLK	OSLK	HALTED	SR	R	SPD	PU
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DLK**: OS double lock status
0: OS double lock not set
1: OS double lock set

Bit 5 **OSLK**: OS lock status
0: OS lock not set
1: OS lock set

Bit 4 **HALTED**: Halted state
0: Processor is in non-debug state
1: Processor is in debug state

Bit 3 **SR**: Sticky reset
0: The non-debug logic of the processor has not been reset since the last time this register was read.
1: The non-debug logic of the processor has been reset since the last time this register was read.

Bit 2 **R**: Reset
0: The non-debug logic is not currently held in reset
1: The non-debug logic is currently held in reset

Bit 1 **SPD**: Sticky power-down status
0: The core power domain has not powered down since the last time this register was read
1: The core power domain has powered down since the last time this register was read

Bit 0 **PU**: Power-up status
0: The core power domain is powered down
1: The core power domain is powered up

DBG processor main ID register (DBG_MIDR)

Address offset: 0xD00

Reset value: 0x410F C075

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMPLEMENTER[7:0]								VARIANT[3:0]				ARCH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PART[11:0]												REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **IMPLEMENTER[7:0]**: Implementer code
0x41: Arm® Limited

Bits 23:20 **VARIANT[3:0]**: Variant (major revision number)
0x0: r0pn

Bits 19:16 **ARCH[3:0]**: Architecture
0xF: Arm® v7

Bits 15:4 **PART[11:0]**: Primary part number
0xC07: Cortex-A7 MPCore

Bits 3:0 **REVISION[3:0]**: Revision (minor revision number)
0x5: rnp5

DBG processor cache type register (DBG_CTR)

Address offset: 0xD04

Reset value: 0x8444 8003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FORMAT[2:0]			Res.	CWG[3:0]				ERG[3:0]				DMINLINE[3:0]			
r	r	r		r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L1IP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IMINLINE[3:0]			
r	r											r	r	r	r

Bits 31:29 **FORMAT[2:0]**: CTR format
0x4: Arm®v7 format

Bit 28 Reserved, must be kept at reset value.

Bits 27:24 **CWG[3:0]**: Cache write-back granule
Number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified.
0x4: 16 words

Bits 23:20 **ERG[3:0]**: Exclusives reservation granule.
Number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions.
0x4: 16 words

Bits 19:16 **DMINLINE[3:0]**: Minimum data cache line.
Number of words in the smallest cache line of all the data and unified caches that the processor controls.
0x4: 16 words

Bits 15:14 **L1IP[1:0]**: L1 instruction cache policy
Indicates the indexing and tagging policy for the L1 instruction cache.
0x2: Virtually Indexed Physically Tagged (VIPT)

Bits 13:4 Reserved, must be kept at reset value.

Bits 3:0 **IMINLINE[3:0]**: Minimum instruction cache line.
Number of words in the smallest cache line of all the instruction caches that the processor controls.
0x3: 8 words

DBG processor TLB type register (DBG_TLBTR)

Address offset: 0xD0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NU
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **NU**: Unified TLB.
0: Processor has a unified TLB

DBG processor multi-processor affinity register (DBG_MPIDR)

Address offset: 0xD14

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	U	Res.	Res.	Res.	Res.	Res.	MT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	r						r								
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	CLUSTERID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	CPUID[1:0]	
				r	r	r	r							r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **U**: Uniprocessor system
0: Processor is part of a multi-processor system

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **MT**: Multi-threading.
Indicates whether the lowest level of affinity consists of logical processors that are implemented using a multi-threading type approach.
0: Processor is part of a multi-processor system

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **CLUSTERID[3:0]**: CLUSTERID configuration.
Indicates the value read in the CLUSTERID configuration pin. It identifies each Cortex-A7 MPCore processor in a system with more than one processor present.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **CPUID[1:0]**: Processor number.
For two processors, the CPU IDs are 0x0 and 0x1.

DBG revision ID register (DBG_REVIDR)

Address offset: 0xD18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ID[31:0]**: Implementation-specific revision information.

DBG processor feature 0 register (DBG_PFR0)

Address offset: 0xD20

Reset value: 0x0000 1131

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STATE3[3:0]				STATE2[3:0]				STATE1[3:0]				STATE0[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **STATE3[3:0]**: Support for Thumb execution environment (ThumbEE) instruction set.
1: ThumbEE instruction set implemented

Bits 11:8 **STATE2[3:0]**: Support for Jazelle extension.
1: Processor supports trivial implementation of Jazelle extension

Bits 7:4 **STATE1[3:0]**: Support for Thumb instruction set.
3: Processor supports Thumb encoding after the introduction of Thumb-2 technology, and for all 16-bit and 32-bit Thumb basic instructions.

Bits 3:0 **STATE0[3:0]**: Support for Arm® instruction set.
1: Processor supports Arm® instruction set

DBG processor feature 1 register (DBG_PFR1)

Address offset: 0xD24

Reset value: 0x0001 1011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GENTIMEXT[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VIRTEXT[3:0]				MPPM[3:0]				SECEXT[3:0]				PROGMOD[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **GENTIMEXT[3:0]**: Generic timer.
1: Processor supports generic timer

Bits 15:12 **VIRTEXT[3:0]**: Virtualisation extensions.
1: Processor supports virtualisation extensions

Bits 11:8 **MPPM[3:0]**: M profile programmers model.
0: Processor does not support microprocessor programmers model

Bits 7:4 **SECEXT[3:0]**: Security extensions.
1: Processor supports security extensions

Bits 3:0 **PROGMOD[3:0]**: Programmers model.
1: Processor supports the standard programmers model for Arm®v4 and later

DBG feature 0 register (DBG_DFR0)

Address offset: 0xD28

Reset value: 0x0201 0555

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PMUEXT[3:0]				MPDBGMOD[3:0]				MMTMOD[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTMOD[3:0]				MMDBGMOD[3:0]				CSDBGMOD[3:0]				CDBGMOD[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **PMUEXT[3:0]**: Performance monitor model.
2: Processor supports Performance monitor unit version 2 (PMUv2) architecture

Bits 23:20 **MPDBGMOD[3:0]**: Debug model, M profile.
0: Processor does not support M profile debug architecture

Bits 19:16 **MMTMOD[3:0]**: Memory mapped trace model.
1: Processor supports Arm® trace architecture, with memory mapped access

Bits 15:12 **CTMOD[3:0]**: Coprocessor trace model
0: Processor does not support Arm® trace architecture, with CP14 access

Bits 11:8 **MMDBGMOD[3:0]**: Memory mapped debug model.
5: Processor supports v7.1 debug architecture, with memory mapped access

Bits 7:4 **CSDBGMOD[3:0]**: Coprocessor secure debug model.
1: Processor supports v7.1 debug architecture, with CP14 access

Bits 3:0 **CDBGMOD[3:0]**: Coprocessor debug model.
1: Processor supports v7.1 debug architecture, with CP14 access

DBG processor memory model feature 0 register (DBG_MMFR0)

Address offset: 0xD30

Reset value: 0x1010 1105

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INNERSH[3:0]				FCSE[3:0]				AUXREGS[3:0]				TCM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SHLVLS[3:0]				OUTERSH[3:0]				PMSA[3:0]				VMSA[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **INNERSH[3:0]**: Innermost shareability.
1: Processor implements hardware coherency support

Bits 27:24 **FCSE[3:0]**: Fast context switch extension.
0: Processor does not support FCSE

Bits 23:20 **AUXREGS[3:0]**: Auxiliary registers.
1: Processor supports the auxiliary control register only

Bits 19:16 **TCM[3:0]**: TCM and associated DMA.
0: Processor does not support TCM

Bits 15:12 **SHLVLS[3:0]**: Shareability levels
1: Processor implements two levels of shareability

Bits 11:8 **OUTERSH[3:0]**: Outermost shareability
1: Processor supports hardware coherency

Bits 7:4 **PMSA[3:0]**: Protected memory system architecture.
1: Processor does not support PMSA

Bits 3:0 **VMSA[3:0]**: Virtual memory system architecture.
1: Processor supports: VMSAv7 with support for remapping and the access flag; Privileged execute never (PXN) within the first level descriptors; 64-bit address translation descriptors

DBG processor memory model feature 1 register (DBG_MMFR1)

Address offset: 0xD34

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPRED[3:0]				L1CTSTCLN[3:0]				L1UNIC[3:0]				L1HVDC[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
L1UNICSW[3:0]				L1HVDCSW[3:0]				L1UNICVA[3:0]				L1HVDCVA[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **BCHPRED[3:0]**: Branch predictor management requirements.
4: Branch predictor requires no flushing at any time

Bits 27:24 **L1CTSTCLN[3:0]**: L1 cache test and clean operations supported.
0: Not supported

Bits 23:20 **L1UNIC[3:0]**: L1 unified cache maintenance operations.
0: Not supported

Bits 19:16 **L1HVDC[3:0]**: L1 Harvard cache maintenance operations.
0: Not supported

Bits 15:12 **L1UNICSW[3:0]**: L1 unified cache set/way line maintenance operations.
0: Not supported

Bits 11:8 **L1HVDCSW[3:0]**: L1 Harvard cache set/way line maintenance operations.
0: Not supported

Bits 7:4 **L1UNICVA[3:0]**: L1 unified cache line maintenance operations by MVA
0: Not supported

Bits 3:0 **L1HVDCVA[3:0]**: L1 Harvard cache line maintenance operations by MVA
0: Not supported

DBG processor memory model feature 2 register (DBG_MMFR2)

Address offset: 0xD38

Reset value: 0x0124 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HWACFLG[3:0]				WFISTALL[3:0]				MEMBARR[3:0]				UNIFTLB[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HVDTLB[3:0]				L1HRANGE[3:0]				L1HBGFTCH[3:0]				L1HFGFTCH[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **HWACFLG[3:0]**: Hardware access flag.
0: Not supported

Bits 27:24 **WFISTALL[3:0]**: Wait for interrupt stalling.
1: Processor supports WFI stalling

Bits 23:20 **MEMBARR[3:0]**: Memory barrier.
2: Processor supports: Data synchronisation barrier (DSB); Instruction syncronisation barrier (ISB); Data memory barrier (DMB)

Bits 19:16 **UNIFTLB[3:0]**: Unified TLB maintenance operations.
4: Processor supports: Invalidate all entries in TLB; Invalidate TLB entry by MVA; Invalidate TLB entries by ASID match; Invalidate TLB entries by MVA all ASID; Invalidate unified hyp TLB entry by MVA; Invalidate entire non-secure non-hyp unified TLB; Invalidate entire hyp unified TLB

Bits 15:12 **HVDTLB[3:0]**: Harvard TLB maintenance operations.
0: Not supported

Bits 11:8 **L1HRANGE[3:0]**: L1 Harvard cache maintenance operation range
0: Not supported

Bits 7:4 **L1HBGFTCH[3:0]**: L1 Harvard cache background prefetch operations
0: Not supported

Bits 3:0 **L1HFGFTCH[3:0]**: L1 Harvard cache foreground prefetch operations

DBG processor memory model feature 3 register (DBG_MMFR3)

Address offset: 0xD3C

Reset value: 0x0210 2211

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUPERSECT[3:0]				CMEMSIZE[3:0]				COHWALK[3:0]				Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r	r	r				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MAINTBCST[3:0]				BPMAINT[3:0]				CMAINTSW[3:0]				CMAINTMVA[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **SUPERSECT[3:0]**: Supersection support.
0: Processor supports supersections

Bits 27:24 **CMEMSIZE[3:0]**: Cache physical memory size.
2: Caches support 40-bit memory address

Bits 23:20 **COHWALK[3:0]**: Coherent walk.
1: Updates to the translation tables do not require a clean to the point of unification to ensure visibility by subsequent translation table walks.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 **MAINTBCST[3:0]**: Maintenance broadcast
2: Cache, TLB and branch predictor operations affect structures according to shareability and defined behavior of instructions.

Bits 11:8 **BPMAINT[3:0]**: Branch predictor maintenance.
2: Processor supports: Invalidate entire branch predictor array; Invalidate branch predictor by MVA.

Bits 7:4 **CMAINTSW[3:0]**: Cache maintenance by set/way.
1: Processor supports: Invalidate data cache by set/way; Clean data cache by set/way; Clean and invalidate data cache by set/way.

Bits 3:0 **CMAINTMVA[3:0]**: Cache maintenance by MVA
1: Processor supports: Invalidate data cache by MVA; Clean data cache by MVA; Clean and invalidate data cache by MVA; Invalidate instruction cache by MVA; Invalidate all instruction cache entries.

DBG processor instruction set attribute 0 register (DBG_ISAR0)

Address offset: 0xD40

Reset value: 0x0210 1110

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DIVINST[3:0]				DBGINST[3:0]				COPROINST[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPBRINST[3:0]				BITFLDINST[3:0]				BITCNTINST[3:0]				SWAPINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **DIVINST[3:0]**: Divide instructions.
2: Processor supports: SDIV and UDIV in the Thumb instruction set; SDIV and UDIV in the Arm® instruction set.

Bits 23:20 **DBGINST[3:0]**: Debug instructions.
1: Processor supports BKPT instruction

Bits 19:16 **COPROINST[3:0]**: Coprocessor instructions.
0: None supported, except for separately attributed architectures including CP15, CP14, and Advanced SIMD and VFP.

Bits 15:12 **CMPBRINST[3:0]**: Combined compare and branch instructions.
1: Processor supports CBNZ and CBZ instructions

Bits 11:8 **BITFLDINST[3:0]**: Bit field instructions.
1: Processor supports BFC, BFI, SBFX and UBFX instructions

Bits 7:4 **BITCNTINST[3:0]**: Bit count instructions.
1: Processor supports CLZ instruction

Bits 3:0 **SWAPINST[3:0]**: Swap instructions.
0: SWP and SWPB instructions supported fractionally

DBG processor instruction set attribute 1 register (DBG_ISAR1)

Address offset: 0xD44

Reset value: 0x1311 2111

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JAZINST[3:0]				INTWKINST[3:0]				IMMINST[3:0]				IFTHENINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXTNDINST[3:0]				EXCARINST[3:0]				EXCINST[3:0]				ENDIANINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **JAZINST[3:0]**: Jazelle extension instructions.
1: Processor supports: BXJ instruction, and the J bit in the PSR

Bits 27:24 **INTWKINST[3:0]**: Interworking instructions.
3: Processor supports: BX instruction, and the T bit in the PSR; BLX instruction, and PC loads have BX-like behavior; Data-processing instructions in the Arm® instruction set with the PC as the destination and the S bit cleared to 0, have BX-like behavior

Bits 23:20 **IMMINST[3:0]**: Immediate instructions.
1: Processor supports: MOVT instruction; MOV instruction encodings with zero-extended 16-bit immediates; Thumb ADD and SUB instruction encodings with zero-extended 12-bit immediates, and other ADD, ADR, and SUB encodings cross-referenced by the pseudocode for those encodings.

Bits 19:16 **IFTHENINST[3:0]**: Thumb If-then instructions.
1: Processor supports the IT instructions, and the IT bits in the PSRs

Bits 15:12 **EXTNDINST[3:0]**: Extend instructions.
2: Processor supports: SXTB, SXTH, UXTB, and UXTH instructions; SXTB16, SXTAB, SXTAB16, SXTAH, UXTB16, UXTAB, UXTAB16, and UXTAH instructions

Bits 11:8 **EXCARINST[3:0]**: A and R profile exception-handling instructions.
1: Processor supports SRS, RFE and CPS instructions

Bits 7:4 **EXCINST[3:0]**: Exception-handling instructions.
1: Processor supports LDM (exception return), LDM (user registers), and STM (user registers) instructions

Bits 3:0 **ENDIANINST[3:0]**: Endian instructions.
1: Processor supports SETEND instruction, and the E bit in the PSRs

DBG processor instruction set attribute 2 register (DBG_ISAR2)

Address offset: 0xD48

Reset value: 0x2123 2041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVSALINST[3:0]				PSRINST[3:0]				MULTUINST[3:0]				MULTSINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MULTINST[3:0]				MULACINST[3:0]				MEMHTINST[3:0]				LDSTRINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REVSALINST[3:0]**: Reversal instructions.
2: Processor supports REV, REV16, REVSH and RBIT instructions

Bits 27:24 **PSRINST[3:0]**: PSR instructions.
1: Processor supports MRS and MSR instructions, and the exception return forms of data-processing instructions

Bits 23:20 **MULTUINST[3:0]**: Advanced unsigned multiply instructions.
2: Processor supports UMULL, UMLAL, and UMAAL instructions

Bits 19:16 **MULTSINST[3:0]**: Advanced signed multiply instructions.

3: Processor supports: SMULL and SMLAL instructions; SMLABB, SMLABT, SMLALBB,SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, SMULWT instructions, and the Q bit in the PSRs; SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD, SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX instructions

Bits 15:12 **MULTINST[3:0]**: Additional multiply instructions.

2: Processor supports MUL, MLA and MLS instructions

Bits 11:8 **MULACINST[3:0]**: Interruptible multi-access instructions.

0: None supported. This means that the processor does not support interruptible LDM and STM instructions

Bits 7:4 **MEMHTINST[3:0]**: Memory hint instructions.

4: Processor supports PLD, PLI (NOP), and PLDW instructions

Bits 3:0 **LDSTRINST[3:0]**: Additional load/store instructions.

1: Processor supports LDRD and STRD instructions

DBG processor instruction set attribute 3 register (DBG_ISAR3)

Address offset: 0xD4C

Reset value: 0x1111 2131

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEEXTINST[3:0]				TRNOPINST[3:0]				THCPYINST[3:0]				TABBRINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYNPRIINST[3:0]				SVCINST[3:0]				SIMDINST[3:0]				SATINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **TEEXTINST[3:0]**: Thumb Execution Environment (ThumbEE) instructions.

1: Processor supports ENTERX and LEAVEX instructions, and modifies the load behavior to include null checking

Bits 27:24 **TRNOPINST[3:0]**: True NOP instructions.

1: Processor supports true NOP instructions in both the Arm® and Thumb instruction sets, and the capability for additional NOP-compatible hints

Bits 23:20 **THCPYINST[3:0]**: Thumb copy instructions.

1: Processor supports Thumb instruction set encoding T1 of the MOV (register) instruction, copying from a low register to a low register

Bits 19:16 **TABBRINST[3:0]**: Table branch instructions.

1: Processor supports TBB and TBH instructions

Bits 15:12 **SYNPRIINST[3:0]**: Synchronisation primitive instructions.

2: Processor supports: LDREX and STREX instructions; CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions; LDREXD and STREXD instructions

Bits 11:8 **SVCINST[3:0]**: SVC instructions.

1: Processor supports SVC instruction

Bits 7:4 **SIMDINST[3:0]**: Single instruction multiple data (SIMD) instructions.

3: Processor supports: SSAT and USAT instructions, and the Q bit in the PSRs; PKHBT, PKHTB, QADD16, QADD8, QASX, QSUB16, QSUB8, QSAX, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSUB16, SHSUB8, SHSAX, SSAT16, SSUB16, SSUB8, SSAX, SXTAB16, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSUB16, UHSUB8, UHSAX, UQADD16, UQADD8, UQASX, UQSUB16, UQSUB8, UQSAX, USAD8, USADA8, USAT16, USUB16, USUB8, USAX, UXTAB16, UXTB16 instructions, and the GE[3:0] bits in the PSRs

Bits 3:0 **SATINST[3:0]**: Saturate instructions.

1: Processor supports QADD, QDADD, QDSUB, QSUB and the Q bit in the PSRs

DBG processor instruction set attribute 4 register (DBG_ISAR4)

Address offset: 0xD50

Reset value: 0x1001 1142

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWPFRAC[3:0]				PSRMINST[3:0]				SPINSFRAC[3:0]				BARRINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMCINST[3:0]				WRBKINST[3:0]				WSHFTINST[3:0]				UNPRIVINST[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **SWPFRAC[3:0]**: Support for the memory system locking the bus for SWP or SWPB instructions:.

1: Processor supports SWP and SWPB instruction but only in a uniprocessor context. SWP and SWPB do not guarantee whether memory accesses from other masters can come between the load memory access and the store memory access of the SWP or SWPB instruction

Bits 27:24 **PSRMINST[3:0]**: M profile instructions to modify the PSRs.

0: None supported

Bits 23:20 **SPINSFRAC[3:0]**: Synchronisation primitive instructions.

0: Processor supports: LDREX and STREX instructions; CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions; LDREXD and STREXD instructions

Bits 19:16 **BARRINST[3:0]**: Barrier instructions.

1: Processor supports DMB, DSB and ISB instructions

Bits 15:12 **SMCINST[3:0]**: SMC instructions.

1: Processor supports SMC instruction

Bits 11:8 **WRBKINST[3:0]**: Write-back addressing modes.
1: Processor supports all Write-Back addressing modes defined in Arm®v7 architecture

Bits 7:4 **WSHFTINST[3:0]**: Instructions with shifts.
4: Processor supports: Shifts of loads and stores over the range LSL 0-3; Constant shift options, both on load/store and other instructions; Register-controlled shift options.

Bits 3:0 **UNPRIVINST[3:0]**: Unprivileged instructions.
2: Processor supports: LDRBT, LDRT, STRBT, and STRT instructions; LDRHT, LDRSBT, LDRSHT, and STRHT instructions.

DBG claim tag set register (DBG_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SET[7:0]**: Set claim tag bits
Write:
00000000: No effect
xxxxxxx1: Set bit 0
xxxxxx1x: Set bit 1
xxxxx1xx: Set bit 2
...etc.
Read:
0xFF: Indicates there are eight bits in claim tag

DBG claim tag clear register (DBG_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CLR[7:0]**: Reset claim tag bits
Write:
00000000: No effect
xxxxxxx1: Clear bit 0
xxxxxx1x: Clear bit 1
xxxxx1xx: Clear bit 2
xxxx1xxx: Clear bit 3
...etc.
Read: Returns current value of claim tag

DBG lock access register (DBG_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: DBG register write access
Enables write access to some DBG registers by processor cores (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access
Other values: Disable write access

DBG lock status register (DBG_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **NTT**: Size of the DBG_LAR register
0: 32-bit

Bit 1 **SLK**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **SLI**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
0: No lock control mechanism exists
1: Lock control mechanism is implemented

DBG authentication status register (DBG_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNI	SNE	SI	SE	NSNI	NSNE	NSI	NSE
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SNI**: Secure non-invasive debug
1: Secure non-invasive debug features implemented

Bit 6 **SNE**: Secure non-invasive debug enabled
0: Non-invasive debug is not permitted in secure PL1 modes
1: Non-invasive debug is permitted in secure PL1 modes

Bit 5 **SI**: Secure invasive debug
1: Secure invasive debug features implemented

Bit 4 **SE**: Secure invasive debug enabled
0: Invasive halting debug is not permitted in secure PL1 modes
1: Invasive halting debug is permitted in secure PL1 modes

Bit 3 **NSNI**: Non-secure non-invasive debug
1: Non-secure non-invasive debug features implemented

Bit 2 **NSNE**: Non-secure non-invasive debug enabled
0: Non-secure non-invasive debug is not permitted
1: Non-secure non-invasive debug is permitted

Bit 1 **NSI**: Non-secure invasive debug
1: Non-secure invasive debug features implemented

Bit 0 **NSE**: Non-secure invasive debug enabled
0: Non-secure invasive debug is not permitted
1: Non-secure invasive debug is permitted

DBG CoreSight device ID 1 register (DBG_DEVID1)

Address offset: 0xFC4

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSROFFSET[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PCSROFFSET[7:0]**: DBGPCSR offset
0x1: No offset applied

DBG Coresight device configuration register (DBG_DEVID)

Address offset: 0xFC8

Reset value: 0x0111 0F13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CIDMASK[3:0]				AUXREGS[3:0]				DOUBLELOCK[3:0]				VIREXTNS[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VECTORCATCH[3:0]				BPADDRMASK[3:0]				WPADDRMASK[3:0]				PCSAMPLE[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **CIDMASK[3:0]**: Context ID matching breakpoint masking
0x0: Not implemented

Bits 27:24 **AUXREGS[3:0]**: Debug external auxiliary control register.
0x1: The processor supports debug external auxiliary control register

Bits 23:20 **DOUBLELOCK[3:0]**: Debug OS double lock register.
0x1: The processor supports debug OS double lock register

Bits 19:16 **VIREXTNS[3:0]**: Virtualisation extensions.
0x1: The processor implements the Virtualization Extensions to the Debug architecture

Bits 15:12 **VECTORCATCH[3:0]**: Vector catch event.
0x0: The processor implements address matching form of vector catch

Bits 11:8 **BPADDRMASK[3:0]**: Immediate Virtual Address (IVA) matching breakpoint masking.
0xF: Not implemented

Bits 7:4 **WPADDRMASK[3:0]**: DVA matching watchpoint masking.
0x1: Watchpoint address mask implemented

Bits 3:0 **PCSAMPLE[3:0]**: Program Counter sampling using debug registers 40, 41, and 42.
0x3: DBGPCSR, DBGCIDSR and DBGVIDSR are implemented as debug registers 40, 41 and 42

DBG CoreSight device type register (DBG_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x5: Debug logic

DBG CoreSight peripheral identity register 4 (DBG_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DBG CoreSight peripheral identity register 0 (DBG_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0007

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x07: DBG part number

DBG CoreSight peripheral identity register 1 (DBG_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0xC: DBG part number

DBG CoreSight peripheral identity register 2 (DBG_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 005B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x5: r0p5

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

DBG CoreSight peripheral identity register 3 (DBG_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DBG CoreSight component identity register 0 (DBG_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

DBG CoreSight component identity register 1 (DBG_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: Debug component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

DBG CoreSight component identity register 2 (DBG_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

DBG CoreSight component identity register 3 (DBG_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Debug unit registers map

Table 590. Debug unit register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DBG_DIDR**	WRPS[3:0]				BRPS[3:0]				CTX_CMPS [3:0]				VERSION [3:0]				DEVIDIMP	NSUHDIMP	PCSRIMP	SEIMP.	Res.	Res.	Res.	Res.	VARIANT [3:0]				REVISION [3:0]			
	Reset value	0	0	1	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	1	1					0	0	0	0	0	1	0	1
0x018	**DBG_WFAR**	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	**DBG_VCR**	NSF	NSI	Res.	NSD	NSP	NSC	NSU	Res.	NSHF	NSHI	NSHE	NSHD	NSHP	NSHC	NSHU	Res.	MF	MI	Res.	MD	MP	MS	Res.	Res.	SF	SI	Res.	SD	SP	SS	SU	R
	Reset value	0	0		0	0	0	0		0	0	0	0	0	0	0		0	0		0	0	0			0	0		0	0	0	0	0

Table 590. Debug unit register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	DBG_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSUCE
	Reset value																																0
0x080	DBG_DTRRX	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	DBG_ITR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	DBG_DSCR	Res.	RXFULL	TXFULL	Res.	RXFULL_L	TXFULL_L	PIPEADV	INSTRCOMPL_L	Res.	Res.	EXTDCCMODE[1:0]		ADADISCARD	NS	SPNIDDIS	SPIDDIS	MDBGEN	HDBGEN	ITREN	UDCCDIS	INTDIS	DBGACK	FS	UND_I	ADABORT_I	SDABORT_I	MOE[3:0]				RESTARTED	HALTED
	Reset value		0	0		0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	DBG_DTRTX	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	DBG_DRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBRRQ	CSPA	CSE	RRQ	HRQ
	Reset value																												0	0	0	0	0
0x094	DBG_DEACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDRSTAT	Res.	Res.	Res.
	Reset value																													0			
0x0A0	DBG_PCSR	PCS[31:1]																															T
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	DBG_CIDSR	CONTEXTIDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	DBG_VIDSR	NS	H	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7:0]							
	Reset value	0	0																							0	0	0	0	0	0	0	0
0x100	DBG_BVR0	ADDRESSCONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	DBG_BVR1	ADDRESSCONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	DBG_BVR2	ADDRESSCONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	DBG_BVR3	ADDRESSCONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110	DBG_BVR4	ADDRESSCONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	DBG_BVR5	ADDRESSCONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 590. Debug unit register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x140	DBG_BCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC [1:0]		E
	Reset value									0	0	0	0	0	0	0	0	0	0	0					0	0	0	0			0	0	0
0x144	DBG_BCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC [1:0]		E
	Reset value									0	0	0	0	0	0	0	0	0	0	0					0	0	0	0			0	0	0
0x148	DBG_BCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC [1:0]		E
	Reset value									0	0	0	0	0	0	0	0	0	0	0					0	0	0	0			0	0	0
0x14C	DBG_BCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC [1:0]		E
	Reset value									0	0	0	0	0	0	0	0	0	0	0					0	0	0	0			0	0	0
0x150	DBG_BCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC [1:0]		E
	Reset value									0	0	0	0	0	0	0	0	0	0	0					0	0	0	0			0	0	0
0x154	DBG_BCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BT[3:0]				LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				Res.	Res.	PMC [1:0]		E
	Reset value									0	0	0	0	0	0	0	0	0	0	0					0	0	0	0			0	0	0
0x180	DBGWVR0	ADDRESS[31:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x184	DBGWVR1	ADDRESS[31:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x188	DBGWVR2	ADDRESS[31:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18C	DBGWVR3	ADDRESS[31:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x1C0	DBG_WCR0	Res.	Res.	Res.	MASK[4:0]					Res.	Res.	Res.	WT	LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				LSC [1:0]		PAC [1:0]		E
	Reset value				0	0	0	0	0				0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x1C4	DBG_WCR1	Res.	Res.	Res.	MASK[4:0]					Res.	Res.	Res.	WT	LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				LSC [1:0]		PAC [1:0]		E
	Reset value				0	0	0	0	0				0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x1C8	DBG_WCR2	Res.	Res.	Res.	MASK[4:0]					Res.	Res.	Res.	WT	LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				LSC [1:0]		PAC [1:0]		E
	Reset value				0	0	0	0	0				0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0
0x1CC	DBG_WCR3	Res.	Res.	Res.	MASK[4:0]					Res.	Res.	Res.	WT	LBN[3:0]				SSC [1:0]		HMC	Res.	Res.	Res.	Res.	BAS[3:0]				LSC [1:0]		PAC [1:0]		E
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x250	DBG_BXVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7::0]							
	Reset value																									0	0	0	0	0	0	0	0
0x254	DBG_BXVR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7::0]							
	Reset value																									0	0	0	0	0	0	0	0
0x300	DBG_OSLAR	OSLOCKACCESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 590. Debug unit register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x304	**DBG_OSLSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSLM[1]	NTT	OSLK	OSLM[0]
	Reset value																													0	0	0	0
0x310	**DBG_PRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COREPURQ	HCWR	CWRR	CORENPDRQ
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	**DBG_PRSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLK	OSLK	HALTED	SR	R	SPD	PU
	Reset value																										0	0	0	0	0	0	0
0xD00	**DBG_MIDR**	IMPLEMENTER[7:0]								VARIANT [3:0]				ARCH[3:0]				PART[11:0]												REVISION [3:0]			
	Reset value	0	1	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	0	1	0	1
0xD04	**DBG_CTR**	FORMAT [2:0]			Res.	CWG[3:0]				ERG[3:0]				DMINLINE [3:0]				L1IP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IMINLINE [3:0]			
	Reset value	1	0	0		0	1	0	0	0	1	0	0	0	1	0	0	1	0											0	0	1	1
0xD0C	**DBG_TLBTR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NU
	Reset value																																0
0xD14	**DBG_MPIDR**	Res.	U	Res.	Res.	Res.	Res.	Res.	MT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLUSTERID [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	CPUID [1:0]	
	Reset value		0						0													0	0	0	0							0	0
0xD18	**DBG_REVIDR**	ID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD20	**DBG_PFR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STATE3[3:0]				STATE2[3:0]				STATE1[3:0]				STATE0[3:0]			
	Reset value																	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	1
0xD24	**DBG_PFR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GENTIMEXT [3:0]				VIRTEXT [3:0]				MPPM [3:0]				SECEXT [3:0]				PROGMOD [3:0]			
	Reset value													0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1
0xD28	**DBG_DFR0**	Res.	Res.	Res.	Res.	PMUEXT [3:0]				MPDBGMOD[3:0]				MMTMOD [3:0]				CTMOD [3:0]				MMDBGMOD [3:0]				CSDBGMOD [3:0]				CDBGMOD [3:0]			
	Reset value					0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1
0xD30	**DBG_MMFR0**	INNERSH [3:0]				FCSE [3:0]				AUXREGS [3:0]				TCM [3:0]				SHLVLS [3:0]				OUTERSH [3:0]				PMSA [3:0]				VMSA [3:0]			
	Reset value	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	1
0xD34	**DBG_MMFR1**	BCHPRED [3:0]				L1CTSTCLN [3:0]				L1UNIC[3:0]				L1HVDC[3:0]				L1UNICSW [3:0]				L1HVDCSW [3:0]				L1UNICVA [3:0]				L1HVDCVA [3:0]			
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD38	**DBG_MMFR2**	HWACFLG [3:0]				WFISTALL [3:0]				MEMBARR [3:0]				UNIFTLB [3:0]				HVDTLB [3:0]				L1HRANGE [3:0]				L1HBGFTCH [3:0]				L1HFGFTCH[3:0]			
	Reset value	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD3C	**DBG_MMFR3**	SUPERSECT[3:0]				CMEMSIZE [3:0]				COHWALK [3:0]				Res.	Res.	Res.	Res.	MAINTBCST [3:0]				BPMAINT [3:0]				CMAINTSW [3:0]				CMAINTMVA [3:0]			
	Reset value	0	0	0	0	0	0	1	0	0	0	0	1					0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	1
0xD40	**DBG_ISAR0**	Res.	Res.	Res.	Res.	DIVINST [3:0]				DBGINST [3:0]				COPROINST [3:0]				CMPBRINST [3:0]				BITFLDINST [3:0]				BITCNTINST [3:0]				SWAPINST [3:0]			
	Reset value					0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0

Table 590. Debug unit register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD44	DBG_ISAR1	JAZINST [3:0]				INTWKINST [3:0]				IMMINST [3:0]				IFTHENINST [3:0]				EXTNDINST [3:0]				EXCARINST [3:0]				EXCINST [3:0]				ENDIANINST[3:0]			
	Reset value	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	1
0xD48	DBG_ISAR2	REVSALINST [3:0]				PSRMINST [3:0]				MULTUINST [3:0]				MULTSINST [3:0]				MULTINST [3:0]				MULACINST [3:0]				MEMHTINST[3:0]				LDSTRINST [3:0]			
	Reset value	0	0	1	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1
0xD4C	DBG_ISAR3	TEEXTINST [3:0]				TRNOPINST [3:0]				THCPYINST [3:0]				TABBRINST [3:0]				SYNPRIINST[3:0]				SVCINST [3:0]				SIMDINST [3:0]				SATINST [3:0]			
	Reset value	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	1	0	0	1	1	0	0	0	1
0xD50	DBG_ISAR4	SWPFRAC [3:0]				PSRMINST [3:0]				SPINSFRAC [3:0]				BARRINST [3:0]				SMCINST [3:0]				WRBKINST [3:0]				WSHFTINST [3:0]				UNPRIVINST[3:0]			
	Reset value	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0
0xFA0	DBG_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFA4	DBG_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFB0	DBG_LAR	KEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFB4	DBG_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
	Reset value																														0	1	1
0xFB8	DBG_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNI	SNE	SI	SE	NSNI	NSNE	NSI	NSE
	Reset value																									1	0	1	0	1	0	1	0
0xFC4	DBG_DEVID1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSROFFSET [3:0]			
	Reset value																													0	0	0	1
0xFC8	DBG_DEVID	CIDMASK [3:0]				AUXREGS [3:0]				DOUBLELOCK [3:0]				VIREXTNS [3:0]				VECTORCATCH [3:0]				BPADDRMASK [3:0]				WPADDRMASK [3:0]				PCSAMPLE [3:0]			
	Reset value	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	1	0	0	1	1
0xFCC	DBG_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	0	1	0	1	0	1
0xFD0	DBG_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	DBG_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	DBG_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	DBG_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																

Table 590. Debug unit register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	DBG_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM [7:0]							
	Reset value																									0	0	0	0	0	1	1	1
0xFE4	DBG_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	0
0xFE8	DBG_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	1	0	1	1	0	1	1
0xFEC	DBG_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DBG_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DBG_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	DBG_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DBG_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									1	0	1	1	0	0	0	1

For the debug unit register descriptions, refer to the *Cortex®-A7 MPCore Technical Reference Manual [9]*.

64.11.2 Cortex®-A7 performance monitoring unit (PMU)

The Performance Monitoring Unit contains four 32-bit event counters, and one processor clock cycle counter. The event counters can be configured to increment every time a specified event occurs. Examples of events that can be counted include:

- Cache hit/miss
- Data load/store
- Bus access
- Interrupt exception
- ETM external trigger output
- Snoop transaction
- Software increment

When a counter wraps, it can generate an interrupt on PMUIRQ. This interrupt is connected to the CTI so can halt the processor(s), start or stop trace, etc.

Most of the events are connected to the ETM via the PMUEVENTS bus. This allows the resources of the ETM to be used to filter the events. The filtered events are returned to the PMU via the two EXTOUT outputs of the ETM.

For a list of PMU events and the corresponding line of the PMUEVENTS bus, refer to the *Cortex®-A7 MPCore Technical Reference Manual [9]*.

Performance monitoring unit registers

The Performance Monitoring Unit registers are accessed by the debugger via the debug APB, at address 0xE00D 1000 to 0xE00D 1FFC for the CA7-CPU1 PMU, and 0xE00D 3000 to 0xE00D 3FFC for the CA7-CPU2. The same registers are accessible by software at addresses 0x500D 1000 to 0x500D 1FFC and 0x500D 3000 to 0x500D 3FFC respectively.

Each CPU can also access the registers of its own PMU via CP15 registers.

For the PMU register descriptions, refer to the *Cortex®-A7 MPCore Technical Reference Manual [9]*.

PMU event count x register (PM_XEVCNTRx)

Address offset: 0x000 + 0x4 *x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

Read or write the value of the selected event counter, PMNn, where n = 0..3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PMN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PMN[31:0]**: Event counter value.

PMU cycle count register (PM_CCNTR)

Address offset: 0x07C

Reset value: 0x0000 0000

Holds the value of the processor cycle counter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CCNT[31:0]**: Cycle count.

Cycle count. Depending on the value of PM_CR.D, this field increments either once every processor clock cycle or once every 64 processor clock cycles.

PMU event type select x register (PM_XEVTYPERx)

Address offset: 0x400 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0XXX XXXX

Controls which events increment the corresponding event counter, and in which modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **P**: Privileged execution filtering.
0: Count events when executing at PL1
1: Do not count events when executing at PL1

Bit 30 **U**: Unprivileged execution filtering.
0: Count events when executing at PL0
1: Do not count events when executing at PL0

Bit 29 **NSK**: Non-secure PL1 control bit.
Controls counting when executing in Non-secure state at PL1. The behavior depends on the combined values of the P and NSK bits:
P=NSK: In Non-secure state, count events when executing at PL1
P!=NSK: In Non-secure state, do not count events when executing at PL1

Bit 28 **NSU**: Non-secure unprivileged control bit.
Controls counting in when executing in Non-secure at PL0. The behavior depends on the combined values of the U and NSU bits:
U=NSU: In Non-secure state, count events when executing at PL0
U!=NSU: In Non-secure state, do not count events when executing at PL0

Bit 27 **NSH**: Non-secure PL2 enable.
0: In Non-secure state, do not count events when executing at PL2
1: In Non-secure state, count events when executing at PL2

Bits 26:8 Reserved, must be kept at reset value.

Bits 7:0 **EVTCOUNT[7:0]**: Event to count.
The event number of the event that is counted by the corresponding event counter, PMNn where n = 0..3

PMU event type select 31 register (PM_XEVTYPER31)

Address offset: 0x47C

Reset value: 0x0XXX XXXX

Controls in which modes the cycle counter increments.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **P**: Privileged execution filtering.
0: Count cycles when executing at PL1
1: Do not count cycles when executing at PL1

Bit 30 **U**: Unprivileged execution filtering.
0: Count cycles when executing at PL0
1: Do not count cycles when executing at PL0

Bit 29 **NSK**: Non-secure PL1 control bit.
Controls counting when executing in Non-secure state at PL1. The behavior depends on the combined values of the P and NSK bits:
P=NSK: In Non-secure state, count cycles when executing at PL1
P!=NSK: In Non-secure state, do not count cycles when executing at PL1

Bit 28 **NSU**: Non-secure unprivileged control bit.
Controls counting in when executing in Non-secure at PL0. The behavior depends on the combined values of the U and NSU bits:
U=NSU: In Non-secure state, count cycles when executing at PL0
U!=NSU: In Non-secure state, do not count cycles when executing at PL0

Bit 27 **NSH**: Non-secure PL2 enable.
0: In Non-secure state, do not count cycles when executing at PL2
1: In Non-secure state, count cycles when executing at PL2

Bits 26:0 Reserved, must be kept at reset value.

PMU count enable set register (PM_CNTENSET)

Address offset: 0xC00

Reset value: 0xXXXX XXXX

Enables the cycle counter and event counters, PMN0..3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
												rw	rw	rw	rw

Bit 31 **C**: Cycle counter enable.
0: disabled (read), no effect (write)
1: enabled (read), enable cycle counter (write)

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **P3**: Event counter 3 enable.
0: disabled (read), no effect (write)
1: enabled (read), enable event counter (write)

Bit 2 **P2**: Event counter 2 enable.
0: disabled (read), no effect (write)
1: enabled (read), enable event counter (write)

Bit 1 **P1**: Event counter 1 enable.
0: disabled (read), no effect (write)
1: enabled (read), enable event counter (write)

Bit 0 **P0**: Event counter 0 enable.
0: disabled (read), no effect (write)
1: enabled (read), enable event counter (write)

PMU count enable clear register (PM_CNTENCLR)

Address offset: 0xC20

Reset value: 0xXXXX XXXX

Disables the cycle counter and event counters, PMN0..3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
												rw	rw	rw	rw

Bit 31 **C**: Cycle counter disable.
0: disabled (read), no effect (write)
1: enabled (read), disable cycle counter (write)

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **P3**: Event counter 3 disable.
0: disabled (read), no effect (write)
1: enabled (read), disable event counter (write)

Bit 2 **P2**: Event counter 2 disable.
0: disabled (read), no effect (write)
1: enabled (read), disable event counter (write)

Bit 1 **P1**: Event counter 1 disable.
0: disabled (read), no effect (write)
1: enabled (read), disable event counter (write)

Bit 0 **P0**: Event counter 0 disable.
0: disabled (read), no effect (write)
1: enabled (read), disable event counter (write)

PMU interrupt enable set register (PM_INTENSET)

Address offset: 0xC40

Reset value: 0xXXXX XXXX

Enables generation of interrupt request by the cycle counter and event counters, PMN0..3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
												rw	rw	rw	rw

Bit 31 **C**: Cycle counter interrupt request enable.
0: disabled (read), no effect (write)
1: enabled (read), enable interrupt request on cycle counter overflow (write)

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **P3**: Event counter 3 interrupt request enable.
0: disabled (read), no effect (write)
1: enabled (read), enable interrupt request on event counter overflow (write)

Bit 2 **P2**: Event counter 2 interrupt request enable.
0: disabled (read), no effect (write)
1: enabled (read), enable interrupt request on event counter overflow (write)

Bit 1 **P1**: Event counter 1 interrupt request enable.
0: disabled (read), no effect (write)
1: enabled (read), enable interrupt request on event counter overflow (write)

Bit 0 **P0**: Event counter 0 interrupt request enable.
0: disabled (read), no effect (write)
1: enabled (read), enable interrupt request on event counter overflow (write)

PMU interrupt enable clear register (PM_INTENCLR)

Address offset: 0xC60

Reset value: 0xXXXX XXXX

Disables generation of interrupt requests by the cycle counter and event counters, PMN0..3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
												rw	rw	rw	rw

Bit 31 **C**: Cycle counter interrupt disable.
0: disabled (read), no effect (write)
1: enabled (read), disable interrupt request on cycle counter overflow (write)

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **P3**: Event counter 3 interrupt disable.
0: disabled (read), no effect (write)
1: enabled (read), disable interrupt request on event counter overflow (write)

Bit 2 **P2**: Event counter 2 interrupt disable.
0: disabled (read), no effect (write)
1: enabled (read), disable interrupt request on event counter overflow (write)

Bit 1 **P1**: Event counter 1 interrupt disable.
0: disabled (read), no effect (write)
1: enabled (read), disable interrupt request on event counter overflow (write)

Bit 0 **P0**: Event counter 0 interrupt disable.
0: disabled (read), no effect (write)
1: enabled (read), disable interrupt request on event counter overflow (write)

PMU overflow flag status register (PM_OVSR)

Address offset: 0xC80

Reset value: 0xXXXX XXXX

Holds the status of the overflow bits for the cycle counter and event counters, PMN0..3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
												rw	rw	rw	rw

Bit 31 **C**: Cycle counter overflow.
0: no overflow occurred (read), no effect (write)
1: overflow occurred (read), reset overflow bit (write)

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **P3**: Event counter 3 overflow.
0: no overflow occurred (read), no effect (write)
1: overflow occurred (read), reset overflow bit (write)

Bit 2 **P2**: Event counter 2 overflow.
0: no overflow occurred (read), no effect (write)
1: overflow occurred (read), reset overflow bit (write)

Bit 1 **P1**: Event counter 1 overflow.
0: no overflow occurred (read), no effect (write)
1: overflow occurred (read), reset overflow bit (write)

Bit 0 **P0**: Event counter 0 overflow.
0: no overflow occurred (read), no effect (write)
1: overflow occurred (read), reset overflow bit (write)

PMU software increment register (PM_SWINC)

Address offset: 0xCA0

Reset value: 0xXXXX XXXX

Increments a counter that is configured to count the software increment event, event 0x00.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **P3**: Event counter 3 increment.
0: No effect
1: Increment counter by 1

Bit 2 **P2**: Event counter 2 increment.
0: No effect
1: Increment counter by 1

Bit 1 **P1**: Event counter 1 increment.
0: No effect
1: Increment counter by 1

Bit 0 **P0**: Event counter 0 increment.
0: No effect
1: Increment counter by 1

PMU configuration register (PM_CFGR)

Address offset: 0xE00

Reset value: 0xXXX9 DF04

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEN	Res.	Res.	EX
												r			r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCD	CC	SIZE[5:0]						N[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **UEN**: User-mode enable register.
1: User-mode enable register implemented

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **EX**: Export.
1: Export is supported

Bit 15 **CCD**: Cycle counter divider.
1: Cycle counter divider implemented

Bit 14 **CC**: Cycle counter.
1: Cycle counter implemented

Bits 13:8 **SIZE[5:0]**: Counter size.
0x1F: 32-bit counters

Bits 7:0 **N[7:0]**: Number of event counters.
0x04: Four event counters

PMU control register (PM_CR)

Address offset: 0xE04

Reset value: 0x4107 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMP[7:0]								IDCODE[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
N[4:0]					Res.	Res.	Res.	Res.	Res.	DP	X	D	C	P	E
r	r	r	r	r						rw	rw	rw	w	w	rw

Bits 31:24 **IMP[7:0]**: Implementer code.
0x41: Arm®

Bits 23:16 **IDCODE[7:0]**: Identification code.
0x07: Cortex-A7 MP core

Bits 15:11 **N[4:0]**: Number of event counters.
In Secure state and Hyp mode, this field returns 0x4 that indicates the number of counters implemented.
In Non-secure modes other than Hyp mode, this field reads the value of the processor HDCR.HPMN register field.

Bits 10:6 Reserved, must be kept at reset value.

Bit 5 **DP**: Disable cycle counter in prohibited regions.
0: Counter is enabled in prohibited regions
1: Counter is disabled in prohibited regions

Bit 4 **X**: Export enable.
This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus:
0: Export of events is disabled
1: Export of events is enabled

Bit 3 **D**: Clock divider.
0: Cycle counter counts every clock cycle
1: Cycle counter counts once every 64 clock cycles

Bit 2 **C**: Clock counter reset.
0: No effect
1: Reset cycle counter to 0

Bit 1 **P**: Event counter reset.
0: No effect
1: Reset all event counters to 0

Bit 0 **E**: Enable.
0: All counters, including the cycle counter, are disabled
1: All counters are enabled

PMU user enable register (PM_USERENR)

Address offset: 0xE08

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: User mode access enable.
0: User mode access is disabled
1: User mode access is enabled

PMU common event ID 0 register (PM_CEID0)

Address offset: 0xE20

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EVENT ID29	EVENT ID28	EVENT ID27	EVENT ID26	EVENT ID25	EVENT ID24	EVENT ID23	EVENT ID22	EVENT ID21	EVENT ID20	EVENT ID19	EVENT ID18	EVENT ID17	EVENT ID16
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EVENT ID15	EVENT ID14	EVENT ID13	EVENT ID12	EVENT ID11	EVENT ID10	EVENT ID9	EVENT ID8	EVENT ID7	EVENT ID6	EVENT ID5	EVENT ID4	EVENT ID3	EVENT ID2	EVENT ID1	EVENT ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **EVENTID29**: Event ID 0x1D user mode access enable
Event number 0x1D corresponds to event:
“Bus cycle”.
0: Event implemented
1: Event not implemented

Bit 28 **EVENTID28**: Event ID 0x1C user mode access enable
Event number 0x1C corresponds to event:
“Instruction architecturally executed, condition code check pass, write to TTBR”.
0: Event implemented
1: Event not implemented

Bit 27 **EVENTID27**: Event ID 0x1B user mode access enable
Event number 0x1B corresponds to event:
“Instruction speculatively executed”.
0: Event implemented
1: Event not implemented

Bit 26 **EVENTID26**: Event ID 0x1A user mode access enable
Event number 0x1A corresponds to event
“Local memory error”.
0: Event implemented
1: Event not implemented

Bit 25 **EVENTID25**: Event ID 0x19 user mode access enable
Event number 0x19 corresponds to event:
“Bus access”.
0: Event implemented
1: Event not implemented

Bit 24 **EVENTID24**: Event ID 0x18 user mode access enable
Event number 0x18 corresponds to event:
“Level 2 data cache write-back”.
0: Event implemented
1: Event not implemented

Bit 23 **EVENTID23**: Event ID 0x17 user mode access enable
Event number 0x17 corresponds to event:
“Level 2 data cache refill”.
0: Event implemented
1: Event not implemented

Bit 22 **EVENTID22**: Event ID 0x16 user mode access enable
Event number 0x16 corresponds to event:
“Level 2 data cache access”.
0: Event implemented
1: Event not implemented

Bit 21 **EVENTID21**: Event ID 0x15 user mode access enable
Event number 0x15 corresponds to event:
“Level 1 data cache write-back”.
0: Event implemented
1: Event not implemented

Bit 20 **EVENTID20**: Event ID 0x14 user mode access enable
Event number 0x14 corresponds to event:
“Level 1 instruction cache access”.
0: Event implemented
1: Event not implemented

Bit 19 **EVENTID19**: Event ID 0x13 user mode access enable
Event number 0x13 corresponds to event:
“Data memory access”.
0: Event implemented
1: Event not implemented

Bit 18 **EVENTID18**: Event ID 0x12 user mode access enable
Event number 0x12 corresponds to event:
“Predictable branch speculatively executed”.
0: Event implemented
1: Event not implemented

Bit 17 **EVENTID17**: Event ID 0x11 user mode access enable
Event number 0x11 corresponds to event:
“Cycle”.
0: Event implemented
1: Event not implemented

Bit 16 **EVENTID16**: Event ID 0x10 user mode access enable
Event number 0x10 corresponds to event:
“Mispredicted or not predicted branch speculatively executed”.
0: Event implemented
1: Event not implemented

Bit 15 **EVENTID15**: Event ID 0xF user mode access enable
Event number 0xF corresponds to event
“Instruction architecturally executed, condition code check pass, unaligned load or store”.
0: Event implemented
1: Event not implemented

Bit 14 **EVENTID14**: Event ID 0xE user mode access enable
Event number 0xE corresponds to event:
“Instruction architecturally executed, condition code check pass, procedure return”.
0: Event implemented
1: Event not implemented

Bit 13 **EVENTID13**: Event ID 0xD user mode access enable
Event number 0xD corresponds to event:
“Instruction architecturally executed, immediate branch”.
0: Event implemented
1: Event not implemented

Bit 12 **EVENTID12**: Event ID 0xC user mode access enable
Event number 0xC corresponds to event:
“Instruction architecturally executed, condition code check pass, software change of the PC”.
0: Event implemented
1: Event not implemented

Bit 11 **EVENTID11**: Event ID 0xB user mode access enable
Event number 0xB corresponds to event:
“Instruction architecturally executed, condition code check pass, write to CONTEXTIDR”.
0: Event implemented
1: Event not implemented

Bit 10 **EVENTID10**: Event ID 0xA user mode access enable
Event number 0xA corresponds to event:
“Instruction architecturally executed, condition code check pass, exception return”.
0: Event implemented
1: Event not implemented

Bit 9 **EVENTID9**: Event ID 0x9 user mode access enable
Event number 0x9 corresponds to event:
“Exception taken”.
0: Event implemented
1: Event not implemented

Bit 8 **EVENTID8**: Event ID 0x8 user mode access enable
Event number 0x8 corresponds to event:
“Instruction architecturally executed”.
0: Event implemented
1: Event not implemented

Bit 7 **EVENTID7**: Event ID 0x7 user mode access enable
Event number 0x7 corresponds to event:
“Instruction architecturally executed, condition code check pass, store”.
0: Event implemented
1: Event not implemented

Bit 6 **EVENTID6**: Event ID 0x6 user mode access enable
Event number 0x6 corresponds to event:
“Instruction architecturally executed, condition code check pass, load”.
0: Event implemented
1: Event not implemented

Bit 5 **EVENTID5**: Event ID 0x5 user mode access enable
Event number 0x5 corresponds to event:
“Level 1 data TLB refill”.
0: Event implemented
1: Event not implemented

Bit 4 **EVENTID4**: Event ID 0x4 user mode access enable
Event number 0x4 corresponds to event:
“Level 1 data cache access”.
0: Event implemented
1: Event not implemented

Bit 3 **EVENTID3**: Event ID 0x3 user mode access enable
Event number 0x3 corresponds to event:
“Level 1 data cache refill”.
0: Event implemented
1: Event not implemented

Bit 2 **EVENTID2**: Event ID 0x2 user mode access enable
Event number 0x2 corresponds to event:
“Level 1 instruction TLB refill”.
0: Event implemented
1: Event not implemented

Bit 1 **EVENTID1**: Event ID 0x1 user mode access enable
Event number 0x1 corresponds to event:
“Level 1 instruction cache refill”.
0: Event implemented
1: Event not implemented

Bit 0 **EVENTID0**: Event ID 0x0 user mode access enable
Event number 0x0 corresponds to event:
“Instruction architecturally executed, condition code check pass, software increment”.
0: Event implemented
1: Event not implemented

PMU lock access register (PM_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]:** PMU register write access
Enables write access to some PMU registers by processor cores
0xC5ACCE55: Enable write access
Other values: Disable write access

PMU lock status register (PM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **NTT:** Size of the PM_LAR register
0: 32-bit

Bit 1 SLK: Current status of lock
0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 SLI: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists.
0: No lock control mechanism exists
1: Lock control mechanism is implemented

PMU authentication status register (PM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNI	SNE	SI	SE	NSNI	NSNE	NSI	NSE
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SNI:** Secure non-invasive debug
1: Secure non-invasive debug features implemented

Bit 6 **SNE:** Secure non-invasive debug enabled
0: Non-invasive debug is not permitted in secure PL1 modes
1: Non-invasive debug is permitted in secure PL1 modes

Bit 5 SI: Secure invasive debug
0: Secure invasive debug features not implemented

Bit 4 SE: Secure invasive debug enabled
0: Invasive halting debug is not permitted in secure PL1 modes

Bit 3 **NSNI:** Non-secure non-invasive debug
1: Non-secure non-invasive debug features implemented

Bit 2 **NSNE:** Non-secure non-invasive debug enabled
0: Non-secure non-invasive debug is not permitted
1: Non-secure non-invasive debug is permitted

Bit 1 NSI: Non-secure invasive debug
0: Non-secure invasive debug features not implemented

Bit 0 NSE: Non-secure invasive debug enabled
0: Non-secure invasive debug is not permitted

PMU CoreSight device type register (PM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0016

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]: Device sub-type identifier**
0x1: Processor

Bits 3:0 **MAJORTYPE[3:0]: Device main type identifier**
0x6: Performance monitors

PMU CoreSight peripheral identity register 4 (PM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]: Register file size**
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]: JEP106 continuation code**
0x4: Arm® JEDEC code

PMU CoreSight peripheral identity register 0 (PM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00A7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]: Part number field, bits [7:0]**
0xA7: PMU part number

PMU CoreSight peripheral identity register 1 (PM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 JEP106ID[3:0]: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]: Part number field, bits [11:8]**
0x9: PMU part number

PMU CoreSight peripheral identity register 2 (PMU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 005B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 REVISION[3:0]: Component revision number
0x5: r0p5

Bit 3 **JEDEC: JEDEC assigned value**
1: Designer ID specified by JEDEC

Bits 2:0 JEP106ID[6:4]: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

PMU CoreSight peripheral identity register 3 (PMU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 REVAND[3:0]: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]: Customer modified**
0x0: No customer modifications

PMU CoreSight component identity register 0 (PMU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]: Component ID field, bits [7:0]**
0x0D: Common ID value

PMU CoreSight component identity register 1 (PMU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 CLASS[3:0]: Component ID field, bits [15:12] - component class
0x9: PMU component

Bits 3:0 **PREAMBLE[11:8]: Component ID field, bits [11:8]**
0x0: Common ID value

PMU CoreSight component identity register 2 (PMU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]: Component ID field, bits [23:16]**
0x05: Common ID value

PMU CoreSight component identity register 3 (PMU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]: Component ID field, bits [31:24]**
0xB1: Common ID value

PMU registers map

Table 591. Performance monitoring unit registers map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PM_XEVCNTR0**	PMN[31:0]																															
	Reset value																																
0x004	**PM_XEVCNTR1**	PMN[31:0]																															
	Reset value																																
0x008	**PM_XEVCNTR2**	PMN[31:0]																															
	Reset value																																
0x00C	**PM_XEVCNTR3**	PMN[31:0]																															
	Reset value																																

Table 591. Performance monitoring unit registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x07C	PM_CCNTR	CCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x400	PM_XEVTYPER0	P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTCOUNT[7:0]							
	Reset value	0	0	0	0	0																											
0x404	PM_XEVTYPER1	P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTCOUNT[7:0]							
	Reset value	0	0	0	0	0																											
0x408	PM_XEVTYPER2	P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTCOUNT[7:0]							
	Reset value	0	0	0	0	0																											
0x40C	PM_XEVTYPER3	P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTCOUNT[7:0]							
	Reset value	0	0	0	0	0																											
0x47C	PM_XEVTYPER31	P	U	NSK	NSU	NSH	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0																											
0xC00	PM_CNTENSET	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
	Reset value																																
0xC20	PM_CNTENCLR	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
	Reset value																																
0xC40	PM_INTENSET	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
	Reset value																																
0xC60	PM_INTENCLR	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
	Reset value																																
0xC80	PM_OVSR	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
	Reset value																																
0xCA0	PM_SWINC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P3	P2	P1	P0
	Reset value																																
0xE00	PM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEN	Res.	Res.	EX	CCD	CC	SIZE[5:0]						N[7:0]							
	Reset value													1			0	1	1	0	1	1	1	1	1	0	0	0	0	0	1	0	0
0xE04	PM_CR	IMP[7:0]								IDCODE[7:0]								N[4:0]					Res.	Res.	Res.	Res.	Res.	DP	X	D	C	P	E
	Reset value	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	1	0	1	1						0	0	0	1	0	0
0xE08	PM_USERENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																
0xE20	PM_CEID0	EVENTID[31:0]																															
	Reset value	0	0												1	1	1												1	1			1

Table 591. Performance monitoring unit registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFB0	PM_LAR	KEY[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFB4	PM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
	Reset value																														0	1	1
0xFB8	PM_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNI	SNE	SI	SE	NSNI	NSNE	NSI	NSE
	Reset value																									1	0	0	0	1	0	0	0
0xFCC	PM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE[3:0]			
	Reset value																									0	0	0	1	0	1	1	0
0xFD0	PM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	PM_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	PM_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	PM_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	PM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM [7:0]							
	Reset value																									0	0	0	0	0	1	1	1
0xFE4	PM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	0
0xFE8	PM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106 ID[6:4]		
	Reset value																									0	1	0	1	1	0	1	1
0xFEC	PM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	PM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	PM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	PM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									0	0	0	0	0	1	0	1

Table 591. Performance monitoring unit registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	PM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									1	0	1	1	0	0	0	1

64.11.3 Embedded trace macrocell (ETM)

The Cortex®-A7 ETM is a CoreSight™ component closely coupled to the CPU. There are two ETM units, one for each CA7 core. The ETM generates trace packets that allow the execution of the Cortex®-A7 core to be traced. Both instruction execution and data accesses can be traced.

The ETM receives information from the CPU over the processor trace interface, including:

- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The addresses of memory locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the *Arm® CoreSight™ ETM™-A7 technical reference manual [6]*.

Embedded trace macrocell registers

The ETM registers are accessed by the debugger via the debug APB, at address range 0xE00D C000 to 0xE00D CFFC. They are accessible by software at address range 0x500D C000 to 0x500D CFFC.

For the ETM register descriptions, refer to the *Arm® CoreSight™ ETM™-A7 technical reference manual [6]*.

ETM control register (ETM_CR)

Address offset: 0x000

Reset value: 0x0000 0461

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	VMIDTRCEN	Res.	TSEN	PROCSEL[2:0]			Res.	Res.	Res.	PORTSIZE[3]	DATAONLY	FILTERCPRT	SUPPDATA	PORTMODE[1:0]	
	rw		rw	rw	rw	rw				r	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTXTIDSIZE[1:0]		PORTMODE[2]	CYCLEACC	ETMPORTSEL	ETMPROG	DBGRQCTL	BRANCHOUTPUT	Res.	PORTSIZE[2:0]			DATAACC[1:0]		MONITORCPRT	ETMPDN
rw	rw	rw	rw	rw	rw	rw	rw		r	r	r	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **VMIDTRCEN**: VMID trace enable
0: VMID tracing disabled
1: VMID tracing enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **TSEN**: Timestamping enable
0: Timestamping disabled
1: Timestamping enabled

Bits 27:25 **PROCSEL[2:0]**: Processor select.
0: Select CPU0 to trace
1: Select CPU1 to traceReserved
2: Reserved
3: Reserved

Bits 24:22 Reserved, must be kept at reset value.

Bit 20 **DATAONLY**: Data-only mode
0: Instruction trace enabled
1: Data-only tracing enabled

Bit 19 **FILTERCPRT**: Filter co-processor register transfers
If MONITORCPRT = 0, this bit should be set to 0.
If MONITORCPRT = 1, this bit has the following effect:
0: All CPRT's traced
1: CPRT's only traced when ViewData is active.

Bit 18 **SUPPDATA**: Enable data suppression
0: Do not suppress data
1: Suppress data when FIFO empty count < ETMFFLR.FIFOFULLLVL

Bits 15:14 **CONTXTIDSIZE[1:0]**: Context ID size.
0: No context ID tracing
1: Context ID bits [7:0] traced
2: Context ID bits [15:0] traced
3: Context ID bits [31:0] traced

Bits 13, 17:16 **PORTMODE[2:0]**: Set trace port clocking mode
Setting PORTMODE[2:0] to any value selects dynamic mode.

Bit 12 **CYCLEACC**: Cycle accurate tracing
0: Disabled
1: Enabled. Include a precise cycle count of executed instructions

Bit 11 **ETMPORTSEL**: Control state of ETMEN output
0: ETMEN is low
1: ETMEN is high
This bit must be set to 1 to enable trace output

Bit 10 **ETMPROG**: ETM programming
This bit must be set to 1 before programming the ETM. Tracing is prevented while this bit is set to 1.
0: ETM operational
1: ETM in programming state

Bit 9 **DBGRQCTL**: Debug request control.

0: DBGRQ output is not set on a trigger event.

1: DBGRQ output is set on a trigger event to force the processor into debug state

Bit 8 **BRANCHOUTPUT**: Branch output

0: Branch addresses not traced

1: All branch addresses are output, even if the branch was because of a direct branch instruction. Setting this bit enables reconstruction of the program flow without having access to the memory image of the code being executed

Bit 7 Reserved, must be kept at reset value.

Bits 21, 6:4 **PORTSIZE[3:0]**: Port size

Setting PORTSIZE[3:0] to any value selects 64-bit port size.

Bits 3:2 **DATAACC[1:0]**: Data access tracing mode.

0: No data tracing

1: Trace only the data portion of the access

2: Trace only the address portion of the access

3: Trace both the address and data of the access

Bit 1 **MONITORCPRT**: Monitor co-processor register transfers

0: CPRT's not traced

1: CPRT's traced. If FILTERCPRT = 1, CPRT's are only traced when ViewData is active.

Bit 0 **ETMPDN**: ETM power down

0: ETM tracing is enabled

1: ETM tracing is disabled, and ETM is in low power mode

ETM configuration code register (ETM_CCR)

Address offset: 0x004

Reset value: 0x8D29 4024

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDREG	Res.	Res.	Res.	SWACC	STARTSTOP	CONTXTIDCMP[1:0]		FIFOFULL	Res.	EXTOUTPUTS[1:0]		EXTINPUTS[2:0]			SEQUENCER
r				r	r	r	r	r		r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COUNTERS[2:0]			MMAPDECS[4:0]					DATACOMPS[3:0]				ADCOMPPRS[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **IDREG**: ETM ID register present

1: ETMIDR register is present and defines the ETM architecture version in use

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **SWACC**: Co-processor and memory access

1: Memory-mapped access to registers is supported

Bit 26 **STARTSTOP**: Trace start/stop block present

1: Trace start/stop block is present

Bits 25:24 **CONTXTIDCMP[1:0]**: Number of context ID comparators

0x1: One context ID comparator implemented

Bit 23 **FIFOFULL**: FIFOFULL logic present
0: FIFOFULL logic is not present in the ETM

Bit 22 Reserved, must be kept at reset value.

Bits 21:20 **EXTOUTPUTS[1:0]**: Number of external outputs
0x2: Two external outputs are supported

Bits 19:17 **EXTINPUTS[2:0]**: Number of external inputs
0x4: Four external inputs are supported

Bit 16 **SEQUENCER**: Sequencer present
1: The sequencer is present in the ETM.

Bits 15:13 **COUNTERS[2:0]**: Number of counters
0x2: Two counters are implemented

Bits 12:8 **MMAPDECS[4:0]**: Number of memory map decoders
0x0: No memory map decoder inputs are implemented.

Bits 7:4 **DATACOMPS[3:0]**: Number of data value comparators
0x2: Two data value comparators are implemented.

Bits 3:0 **ADCOMPPRS[3:0]**: Number of address comparator pairs
0x4: Four address comparator pairs are implemented.

ETM trigger register (ETM_TRIGGER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function
If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier
See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM status register (ETM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIG	STSTPRESSTAT	PROGBIT	UNTRCDOVFL
												rw	rw	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **TRIG**: Trigger flag
Set when the trigger occurs, and prevents the trigger from being output until the ETM is programmed again.

Bit 2 **STSTPRESSTAT**: Trigger start/stop resource status
Holds the current status of the trace start/stop resource. If set to 1, it indicates that a trace on address has been matched, without a corresponding trace off address match.

Bit 1 **PROGBIT**: Programming bit value (read-only)
The current effective value of the ETM Programming bit, bit [10] of the ETM_CR. Set the programming bit to 1 and wait for this bit to go to 1 before programming the ETM.
This bit remains at 0 until the FIFO is empty, or if OS lock is set in the ETM_OSLSR register.

Bit 0 **UNTRCDOVFL**: Untraced overflow (read-only)
If set to 1, there is an overflow that has not yet been traced. This bit is cleared to 0 when trace is restarted, or when the ETM power-down bit (bit [0] of the ETM_CR register) is set to 1.

ETM configuration register (ETM_SCR)

Address offset: 0x014

Reset value: 0x0002 0D09

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOFTCHCOMP	Res.
														r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUPPROC[2:0]			PMSUPP	PSSUPP	MAXPS[3]	FFSUPP	Res.	Res.	Res.	Res.	Res.	MAXPS[2:0]		
	r	r	r	r	r	r	r						r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **NOFTCHCOMP**: No fetch comparisons
1: Fetch comparisons are not implemented

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:12 **SUPPROC[2:0]**: Number of supported processors
0x0: One processor supported

Bit 11 **PMSUPP**: Port mode support
0: Current selected port mode is not supported
1: Current selected port mode is supported

Bit 10 **PSSUPP**: Port size support
0: Current selected port size is not supported
1: Current selected port size is supported

Bit 8 **FFSUPP**: FIFOFULL support
0: FIFOFULL is not supported

Bits 7:3 Reserved, must be kept at reset value.

Bits 9, 2, 1, 0 **MAXPS[3:0]**: Maximum ETM port size field
These bits indicate the maximum ETM port size supported.
0x4: Maximum port size = 4
Others: Reserved

ETM trace enable start/stop control register (ETM_SSCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPADDRCOMPSEL[16:1]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STARTADDRCOMPSEL[16:1]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **STOPADDRCOMPSEL[16:1]**: Stop address comparator select bits
When bit n is set to 1, it selects single address comparator n as stop address.

Bits 15:0 **STARTADDRCOMPSEL[16:1]**: Start address comparator select bits
When bit n is set to 1, it selects single address comparator n as start address.

ETM trace enable control 2 register (ETM_TECR2)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INCEXCCTLCOMPSEL[16:1]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INCEXCCTLCOMPSEL[16:1]**: Include/exclude control comparator select bits
When bit n is set to 1, it selects single address comparator n for include/exclude control.

ETM trace enable event register (ETM_TEEVR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM trace enable control 1 register (ETM_TECR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TSSEN	INCEX CCTL	INCEXCMMDSEL[16:9]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INCEXCMMDSEL[8:1]								INCEXCADCOMPSEL[8:1]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TSSEN**: Trace start/stop trace enable control
0: Trace start/stop block does not control trace enable
1: Trace enable is controlled by the trace start/stop block

Bit 24 **INCEXCCTL**: Trace start/stop trace enable control
0: Include. The specified resources indicate the regions where tracing can occur. When outside this region tracing is prevented
1: Exclude. The resources specified in bits 23:0 and in the ETMTECR2 register indicate regions to be excluded from the trace. When outside an exclude region, tracing can occur.

Bits 23:8 **INCEXCMMDSEL[16:1]**: Memory map decode select bits for include/exclude control.
Not used, there are no memory map decoders

Bits 7:0 **INCEXCADCOMPSEL[8:1]**: Address comparator select bits for include/exclude control.
When bit n is set to 1, it selects address range comparator n for include/exclude control

ETM FIFOFULL level register (ETM_FFLR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOFULLLVL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FIFOFULLLVL[7:0]**: Threshold for FIFOFULL signal
The number of bytes left in the FIFO, below which the SuppressData signal is asserted. For example, setting this value to 15 causes data trace suppression, if enabled, when there are less than 15 free bytes in the FIFO.

ETM view data event register (ETM_VDEVR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM view data control 1 register (ETM_VDCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VDEXSADCOMPSEL[16:1]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VDINCSADCOMPSEL[16:1]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **VDEXSADCOMPSEL[16:1]**: Single address comparator select bits for ViewData exclude control

When bit n is set to 1, it selects single address comparator n for exclude control.

Bits 15:0 **VDINCSADCOMPSEL[16:1]**: Single address comparator select bits for ViewData include control

When bit n is set to 1, it selects single address comparator n for include control.

ETM view data control 3 register (ETM_VDCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCONLY
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VDEXCADCOMPSEL[8:1]								VDINCADCOMPSEL[8:1]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **EXCONLY**: Exclude only

0: Mixed mode. ViewData operates in a mixed mode, and both include and exclude resources can be programmed

1: Exclude-only mode. ViewData is programmed onlry in an excluding mode. If none of the excluding resources match, tracing can occur

Bits 15:8 **VDEXCADCOMPSEL[8:1]**: Address range comparator select bits for ViewData exclude control

When bit n is set to 1, it selects address range comparator n for exclude control.

Bits 7:0 **VDINCADCOMPSEL[8:1]**: Address range comparator select bits for ViewData include control

When bit n is set to 1, it selects single address comparator n for include control.

ETM address comparator value x register (ETM_ACVRx)

Address offset: 0x040 + 0x4 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address for comparison

ETM address comparator type x register (ETM_ACTRx)

Address offset: 0x080 + 0x4 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VMID COMP	HYP MODE	STMODE[3:0]				CTXTIDCOMP [1:0]		EXACT MATCH	DCOMP[1:0]		COMPACCSIZE [1:0]		ACCTYPE[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **VMIDCOMP**: Virtual machine ID (VMID) comparison enable
A value of 1 means that the address comparator matches only if the current VMID matches the value stored in the ETM_VMIDCVR register

Bit 14 **HYPMODE**: Hyp mode comparison enable
A value of 1 means that the address comparator also matches if the processor is operating in Hyp mode

Bits 13:10 **STMODE[3:0]**: State and mode comparison control
Bits [3,1] control non-secure state comparison. Bits [2,0] control secure state comparison.
X0X0: Match in all modes in secure state
X0X1: Do not match in any modes in secure state
X1X0: Match in all modes except User mode in secure state
X1X1: Match only in User mode in secure state
0X0X: Match in all modes in non-secure state
0X1X: Do not match in any modes in non-secure state
1X0X: Match in all modes except User mode in non-secure state
1X1X: Match only in User mode in non-secure state

Bits 9:8 **CTXTIDCOMP[1:0]**: Context ID comparator control.
0: Ignore context ID comparator
1: Address comparator matches only if context ID comparator value 1 matches
2: Address comparator matches only if context ID comparator value 2 matches
3: Address comparator matches only if context ID comparator value 3 matches

Bit 7 **EXACTMATCH**: Exact match
0: Comparator matches even for cancelled instructions, data transfer aborts or failed stores
1: Comparator does not match for cancelled instructions, data transfer aborts or failed stores

Bits 6:5 **DCOMP[1:0]**: Data value comparison control.
0: No data value comparison is made
1: Comparator can match only if data value matches
2: Reserved
3: Comparator can match only if data value does not match

Bits 4:3 **COMPACCSIZE[1:0]**: Comparison access size.
0: Java instruction or byte data
1: Thumb instruction or halfword data
2: Reserved
3: Arm instruction or word data

Bits 2:0 **ACCTYPE[2:0]**: Access type
0: Instruction fetch (not supported)
1: Instruction execute
2: Instruction executed and passed condition code test
3: Instruction executed and failed condition code test
4: Data load or store
5: Data load
6: Data store
7: Reserved

ETM data comparator value x register (ETM_DCVRx)

Address offset: 0x0C0 + 0x4 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data value for comparison

ETM data comparator mask x register (ETM_DCMRx)

Address offset: 0x100 + 0x4 * (x - 1), (x = 1 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Data mask
When a data mask bit is set to 1, the corresponding bit in the ETM_DCVRn register is disregarded in the comparison and must be set to zero

ETM counter reload value x register (ETM_CNTRLDVRx)

Address offset: 0x140 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITIALCOUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INITIALCOUNT[15:0]**: Initial count
Specifies the counter reload value.

ETM counter enable x register (ETM_CNTENRx)

Address offset: 0x150 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM counter reload event x register (ETM_CNTRLDEVRx)

Address offset: 0x160 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM counter value x register (ETM_CNTVRx)

Address offset: 0x170 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INITIALCOUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INITIALCOUNT[15:0]**: Initial count
Specifies the counter reload value.

ETM sequencer state 1-2 transition event register (ETM_SQ12EVR)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function
If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier
See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM sequencer state 2-1 transition event register (ETM_SQ21EVR)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM sequencer state 2-3 transition event register (ETM_SQ23EVR)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM sequencer state 3-1 transition event register (ETM_SQ31EVR)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM sequencer state 3-2 transition event register (ETM_SQ32EVR)

Address offset: 0x190

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier
Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM sequencer state 1-3 transition event register (ETM_SQ13EVR)

Address offset: 0x194

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM current sequencer state register (ETM_SQR)

Address offset: 0x19C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQSTATE[1:0]	
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SQSTATE[1:0]**: Current sequencer state

0: state 1
1: state 2
2: state 3
3: Reserved

ETM external output event x register (ETM_EXTOUTEVRx)

Address offset: 0x1A0 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM context ID comparator value 1 register (ETM_CIDCVR1)

Address offset: 0x1B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONTEXTID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTEXTID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CONTEXTID[31:0]**: Context ID value for comparison

ETM context ID comparator mask register (ETM_CIDCMR)

Address offset: 0x1BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MASK[31:0]**: Context ID mask

When a mask bit is set to 1, the corresponding bit in the ETM_CIDCVR1 register is disregarded in the comparison and must be set to zero

ETM synchronization frequency register (ETM_SYNCFR)

Address offset: 0x1E0

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	FREQUENCY[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FREQUENCY[11:0]**: Synchronisation frequency.

This value is the time in bytes between synchronization points in the trace, used by ETM (the tools can start decompressing only at synchronization points).

ETM ID register (ETM_IDR)

Address offset: 0x1E4

Reset value: 0x410C F250

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IMPLEMENTER[7:0]								Res.	Res.	Res.	BPENC	SECEXT	T32	Res.	LDPC1ST
r	r	r	r	r	r	r	r				r	r	r		r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PROCFAM[3:0]				ARCHVMAJ[3:0]				ARCHVMIN[3:0]				REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **IMPLEMENTER[7:0]**: Implementer code
0x41: Arm®

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **BPENC**: Branch packet encoding support
0: Supports the original branch packet encoding

Bit 19 **SECEXT**: Security extensions support
1: Supports security extensions

Bit 18 **T32:** 32-bit Thumb instruction tracing
1: Supports 32-bit Thumb instructions

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LDPC1ST**: Load PC first
0: If an LSM load operation with the PC, PC is not loaded first in the list

Bits 15:12 **PROCFAM[3:0]**: Processor family
0xF: Processor family is not identified in this register

Bits 11:8 **ARCHVMAJ[3:0]**: Major ETM architecture version
0x2: Version 3.x

Bits 7:4 **ARCHVMIN[3:0]**: Minor ETM architecture version
0x5: Version x.5

Bits 3:0 **REVISION[3:0]**: Implementation revision
0x0: Revision r0p0

ETM configuration code extension register (ETM_CCER)

Address offset: 0x1E8

Reset value: 0x3440 08F2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TSSIZE	TSENC	CNTFNC	VIRTEXT	Res.	Res.	Res.	TIMSTMP	ETMEIBCR	TSSUWP	EMBICEWPS[3:0]			
		r	r	r	r				r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSTRES[2:0]			DACOMPS	ARR	PMUEVENTREGS[7:0]								EEISELS[2:0]		
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TSSIZE**: Timestamp size
1: 64 bits.

Bit 28 **TSENC**: Timestamp encoding
1: Timestamp is encoded as a natural binary number

Bit 27 **CNTFNC**: Counter function
0: Counter is a full function counter

Bit 26 **VIRTEXT**: Virtual extensions
1: Virtual extensions supported

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 **TIMSTMP**: Timestamping
1: Timestamping supported

Bit 21 **ETMEIBCR**: EmbeddedICE behavior control implemented
0: Not implemented

Bit 20 **TSSUWP**: Trace start/stop uses Embedded ICE watchpoint inputs
0: No embeddedICE watchpoint inputs supported

Bits 19:16 **EMBICEWPS[3:0]**: Embedded ICE watchpoint inputs
0: No embeddedICE watchpoint inputs supported

Bits 15:13 **INSTRES[2:0]**: Instrumentation resources
0: None

Bit 12 **DACOMPS**: Data address comparison support
0: No data address comparisons are supported

Bit 11 **ARR**: Readable registers
1: All registers are readable

Bits 10:3 **PMUEVENTREGS[7:0]**: Number of PMU event registers available
0x1E: 30 registers

Bits 2:0 **EEISELS[2:0]**: Number of extended external input selectors
0x2: 2 selectors

ETM extended external input selection register (ETM_EXTINSELR)

Address offset: 0x1EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EXTINSEL2[7:0]								EXTINSEL1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **EXTINSEL2[7:0]**: Extended external input selector 2

Bits 7:0 **EXTINSEL1[7:0]**: Extended external input selector 1

ETM timestamp event register (ETM_TSEVR)

Address offset: 0x1F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.

0x00: Single address comparator 1
...
0x0F: Single address comparator 16
0x10: Address range comparator 1
...
0x17: Address range comparator 8
0x18: Instrumentation resource 1
0x19: Instrumentation resource 2
0x1A: Instrumentation resource 3
0x1B: Instrumentation resource 4
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
...
0x27: Embedded-ICE watchpoint comparators 8 (from DWT)
0x40: Counter 1 at zero
0x41: Counter 2 at zero
0x42: Counter 3 at zero
0x43: Counter 4 at zero
0x50: Sequencer in states 1
0x51: Sequencer in states 2
0x52: Sequencer in states 3
0x58: Context ID comparator 1
0x5B: VMID comparator
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x62: External inputs 3
0x63: External inputs 4
0x6D: Processor is in non-secure state
0x6E: Trace prohibited by processor
0x6F: Hard-wired input, always true
Others: Reserved

ETM auxiliary control register (ETM_AUXCR)

Address offset: 0x1FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPKTTRCDIS	TSPKTGENSUP	DNFOVFL
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **DPKTTRCDIS**: Disable tracing of data packets
0: Not disabled
1: Disabled

Bit 1 **TSPKTGENSUP**: Suppress generation of timestamp packets
0: Do not suppress
1: Suppress generation of timestamp packets as the result of executing an ISB instruction

Bit 0 **DNFOVFL**: Do not force FIFO overflow
0: Force FIFO overflow if a periodic synchronisation has not been output for two full synchronisation periods
1: Do not force FIFO overflow

ETM trace ID register (ETM_TRACEIDR)

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRACEID[7:0]**: Trace ID to output onto the trace bus
This should be programmed with a unique value to differentiate it from other trace sources in the system.

ETM ID 2 register (ETM_IDR2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPTO	RFETO
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWPTO**: SWP or SWPB instruction order
Identifies the order for a SWP or SWPB instruction.
0: The Load transfer is traced before the Store transfer

Bit 0 **RFETO**: RFE instruction order
Identifies the order for a RFE instruction.
0: The PC transfer is traced before the CPSR transfer

ETM VMID comparator value register (ETM_VMIDCVR)

Address offset: 0x240

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **VMID[7:0]**: Virtual machine ID for comparison with current one

ETM OS lock access register (ETM_OSLAR)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEYVALUE[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEYVALUE[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEYVALUE[31:0]**: ETM register lock access
0xC5ACCE55: Disable access
Other values: Enable access

ETM OS lock status register (ETM_OSLSR)

Address offset: 0x304

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETM OSLAR	NTT	DBGR LKED	DBGR LKSUP
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **ETMOSLAR**: ETM OS lock access register implemented
1: Implemented

Bit 2 **NTT**: 32-bit access required
0: 32-bit accesses are required to operate the ETM_OSLAR

Bit 1 **DBGRLKED**: Registers locked
0: ETM registers are not locked
1: ETM registers are locked. Any access will return a slave-generated error response

Bit 0 **DBGRLKSUP**: ETM power down support
0: Full support

ETM OS save & restore register (ETM_OSSRR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VALUE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **VALUE[31:0]**: OS save/restore register value
On first read after locking the registers, this register contains the number of additional accesses required to save or restore the ETM registers.
Subsequent reads during an OS save return the contents of each register for storing in memory. During an OS restore, subsequent writes load the written value to each register in turn.

ETM device power-down control register (ETM_PDCR)

Address offset: 0x310

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCPWREN	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **TRCPWREN**: Trace power enable
1: Power to the domain containing the trace registers must be provided.

Bits 2:0 Reserved, must be kept at reset value.

ETM device power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSLOCKST	Res.	Res.	Res.	STKYREGST	ETMPWRD
										r				r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **OSLOCKST**: OS lock status
The value of this bit is the same as the value of bit[1] of the ETM_OSLSR.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STKYREGST**: Sticky register state
0: Indicates this register has been read after a reset
1: Indicates this register has never been read after a reset

Bit 0 **ETMPWRD**: ETM powered
1: ETM powered; trace registers can be accessed.

ETM claim tag set register (ETM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CLAIMSET[7:0]**: Set claim tag bits
Write:
00000000: No effect
xxxxxxx1: Set bit 0
xxxxxx1x: Set bit 1
xxxxx1xx: Set bit 2
...etc.
Read:
0xFF: Indicates there are eight bits in claim tag

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CLAIMCLR[7:0]**: Reset claim tag bits
Write:
00000000: No effect
xxxxxxx1: Clear bit 0
xxxxxx1x: Clear bit 1
xxxxx1xx: Clear bit 2
xxxx1xxx: Clear bit 3
...etc.
Read: Returns current value of claim tag

ETM lock access register (ETM_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETM register write access

Enables write access to some ETM registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access

Other values: Disable write access

ETM lock status register (ETM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK TYPE	LOCK GRANT	LOCK EXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the ETM_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns zero when read by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism

The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists

1: Lock control mechanism is implemented

ETM authentication status register (ETM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x: Unused
10: Secure non-invasive debug disabled
11: Secure non-invasive debug enabled

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
00: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x: Unused
10: Non-secure non-invasive debug disabled
11: Non-secure non-invasive debug enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
00: Not implemented

ETM Coresight device configuration register (ETM_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DEVICEID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DEVICEID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DEVICEID[31:0]**: Device configuration
0x0: No configuration information

ETM CoreSight device type register (ETM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0056

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x56: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETM CoreSight component identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Embedded trace macrocell registers map

Table 592. ETM registers map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ETM_CR	Res.	VMIDTRCEN	Res.	TSEN.	PROCSEL[2:0]			Res.	Res.	Res.	PORTSIZE[3]	DATAONLY	FILTERCPRT	SUPPDATA.	PORTMODE[1:0]		CONTXTIDSIZE[1:0]		PORTMODE[2]	CYCLEACC	ETMPORTSEL.	ETMPROG	DBGRQCTL	BRANCHOUTPUT	Res.	PORTSIZE[2:0]			DATAACC[1:0]		MONITORCPRT	ETMPDN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1
0x004	ETM_CCR	IDREG	Res.	Res.	Res.	SWACC	STARTSTOP	CONTXTIDCMP[1:0]		FIFOFULL	Res.	EXTOUTPUTS[1:0]		EXTINPUTS[2:0]			SEQUENCER	COUNTERS[2:0]			MMAPDECS[4:0]					DATACOMPS[3:0]				ADCOMPPRS[3:0]			
	Reset value	1	0	0	0	1	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0
0x008	ETM_TRIGGER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																																
0x010	ETM_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIG	STSTPRESSTAT	PROGBIT	UNTRCDOVFL
	Reset value																																
0x014	ETM_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOFTCHCMP	Res.	Res.	SUPPROC[2:0]			PMSUPP	PSSUPP	MAXPS[3]	FFSUPP	Res.	Res.	Res.	Res.	Res.	MAXPS[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
0x018	ETM_SSCR	STOPADDRCOMPSEL[16:1]																STARTADDRCOMPSEL[16:1]															
	Reset value																																
0x01C	ETM_TECR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCEXCCTLCOMPSEL[16:1]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	ETM_TEEVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	ETM_TECR1	Res.	Res.	Res.	Res.	Res.	Res.	TSSEN	INCEXCCTL	INCEXCMMDSEL[16:1]																INCEXCADCOMPSEL [8:1]							
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	ETM_FFLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOFULLLVL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 592. ETM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	ETM_VDEVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETM_VDCR1	VDEXCSADCOMPSEL[16:1]																VDINCSADCOMPSEL[16:1]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	ETM_VDCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCONLY	VDEXCADCOMPSEL[8:1]								VDINCADCOMPSEL[8:1]							
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	ETM_ACVR1	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	ETM_ACVR2	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	ETM_ACVR3	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	ETM_ACVR4	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	ETM_ACVR5	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	ETM_ACVR6	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	ETM_ACVR7	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	ETM_ACVR8	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	ETM_ACTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ETM_ACTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 592. ETM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ETM_ACTR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ETM_ACTR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	ETM_ACTR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	ETM_ACTR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098	ETM_ACTR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																																
0x09C	ETM_ACTR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMIDCOMP	HYPMODE	STMODE[3:0]				CTXTIDCOMP[1:0]		EXACTMATCH	DCOMP[1:0]		COMPACCSIZE[1:0]		ACCTYPE		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0	ETM_DCVR1	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8	ETM_DCVR3	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 592. ETM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	ETM_DCMR1	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	ETM_DCMR3	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	ETM_CNTRLDVR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITIALCOUNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	ETM_CNTRLDVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITIALCOUNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	ETM_CNTENR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value															1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	ETM_CNTENR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value															1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x160	ETM_ CNTRLDEVR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164	ETM_ CNTRLDEVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x170	ETM_CNTVR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x174	ETM_CNTVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x180	ETM_SQ12EVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	ETM_SQ21EVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	ETM_SQ23EVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	ETM_SQ31EVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	ETM_SQ32EVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194	ETM_SQ13EVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 592. ETM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x19C	ETM_SQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQSTATE[1:0]	
	Reset value																															0	0
0x1A0	ETM_EXTOUTEVR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A4	ETM_EXTOUTEVR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1B0	ETM_CIDCVR1	CONTEXTID[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1BC	ETM_CIDCMR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1E0	ETM_SYNCFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCFREQ[11:0]											
	Reset value																					0	1	0	0	0	0	0	0	0	0	0	0
0x1E4	ETM_IDR	IMPLEMENTER[7:0]								Res.	Res.	Res.	BPENC	SECEXT	T32	Res.	LDPC1ST	PROCFAM [3:0]				ARCHVMAJ [3:0]				ARCHVMIN [3:0]				REVISION [3:0]			
	Reset value	0	1	0	0	0	0	0	1				0	1	1		0	1	1	1	1	0	0	1	0	0	1	0	1	0	0	0	0
0x1E8	ETM_CCER	Res.	Res.	TSSIZE	TSENC	CNTFNC	VIRTEXT	Res.	Res.	Res.	TIMSTMP	ETMEIBCR	TSSBLK	EMBICEWPS[3:0]				INSTRES[2:0]			DACOMPS	ARR	PMUEVENTREGS[7:0]								EEISELS		
	Reset value			1	1	0	1				1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	1	0	0	1	0
0x1EC	ETM_EXTINSELR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTINSEL2[7:0]								EXTINSEL1[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1F8	ETM_TSEVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2:0]			RESOURCEB[6:0]							RESOURCEA[6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1FC	ETM_AUXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPKTTRCDIS	TSPKTGENSU	DNFOVFL
	Reset value																														0	0	0
0x200	ETM_TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x208	ETM_IDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPTO	RFETO
	Reset value																															0	0

Table 592. ETM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x240	ETM_VMIDCVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VMID[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x300	ETM_OSLAR	KEYVALUE[31:0]																															
	Reset value																																
0x304	ETM_OSLSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETMOSLAR	NTT	DBGRLKED	DBGRLKSU
	Reset value																													1	0	1	0
0x308	ETM_OSSRR	VALUE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x310	ETM_PDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCPWREN	Res.	Res.	Res.
	Reset value																													1			
0x314	ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSLOCKST	Res.	Res.	Res.	STKYREGST	ETMPWRD
	Reset value																											1				1	1
0xFA0	ETM_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET							
	Reset value																									1	1	1	1	1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR							
	Reset value																									0	0	0	0	0	0	0	0
0xFB0	ETM_LAR	KEY																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFB4	ETM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
	Reset value																																
0xFB8	ETM_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID		SID		NSNID		NSID	
	Reset value																									0	0	0	0	1	1	0	0
0xFC8	ETM_DEVID	DEVICEID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFD0	ETM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB TYPE				MAJOR TYPE			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE				JEP106 CON			
	Reset value																									0	0	0	0	0	1	0	0

Table 592. ETM registers map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD4	ETM_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	ETM_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	ETM_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM [7:0]							
	Reset value																									0	1	0	1	0	1	1	0
0xFE4	ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND				CMOD			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS				PREAMBLE			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE							
	Reset value																									1	0	1	1	0	0	0	1

64.11.4 Cross trigger interface (CTI)

See *Section 64.10.3*.

64.12 Cortex®-M4 debug features

The Cortex®-M4 subsystem integrates the following CoreSight™ components:

- CM4 ROM table
- CM4 System Control Space (SCS)
- Breakpoint Unit (FPB)
- Data Watchpoint and Trace Unit (DWT)
- Instrumentation Trace Macrocell (ITM)
- Embedded Trace Macrocell (ETM)
- Cross Trigger Interface (CTI)

These components are accessible by the debugger via the Cortex®-M4 AHB-AP.

64.12.1 ROM tables

The ROM table is a CoreSight™ component that contains the base addresses of all the CoreSight™ debug components accessible via the AHB-AP. These tables allow a debugger to discover the topology of the CoreSight™ system automatically.

There is one ROM table in the Cortex®-M4 subsystem. This table is pointed to by the BASE register in the Cortex®-M4 AHB-AP. It contains the base address pointer for the System Control Space registers, which allow the debugger to identify the CPU core, as well as for the FPB, DWT, ITM, ETM and CTI.

The Cortex®-M4 CPU ROM table described in *Table 593* occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

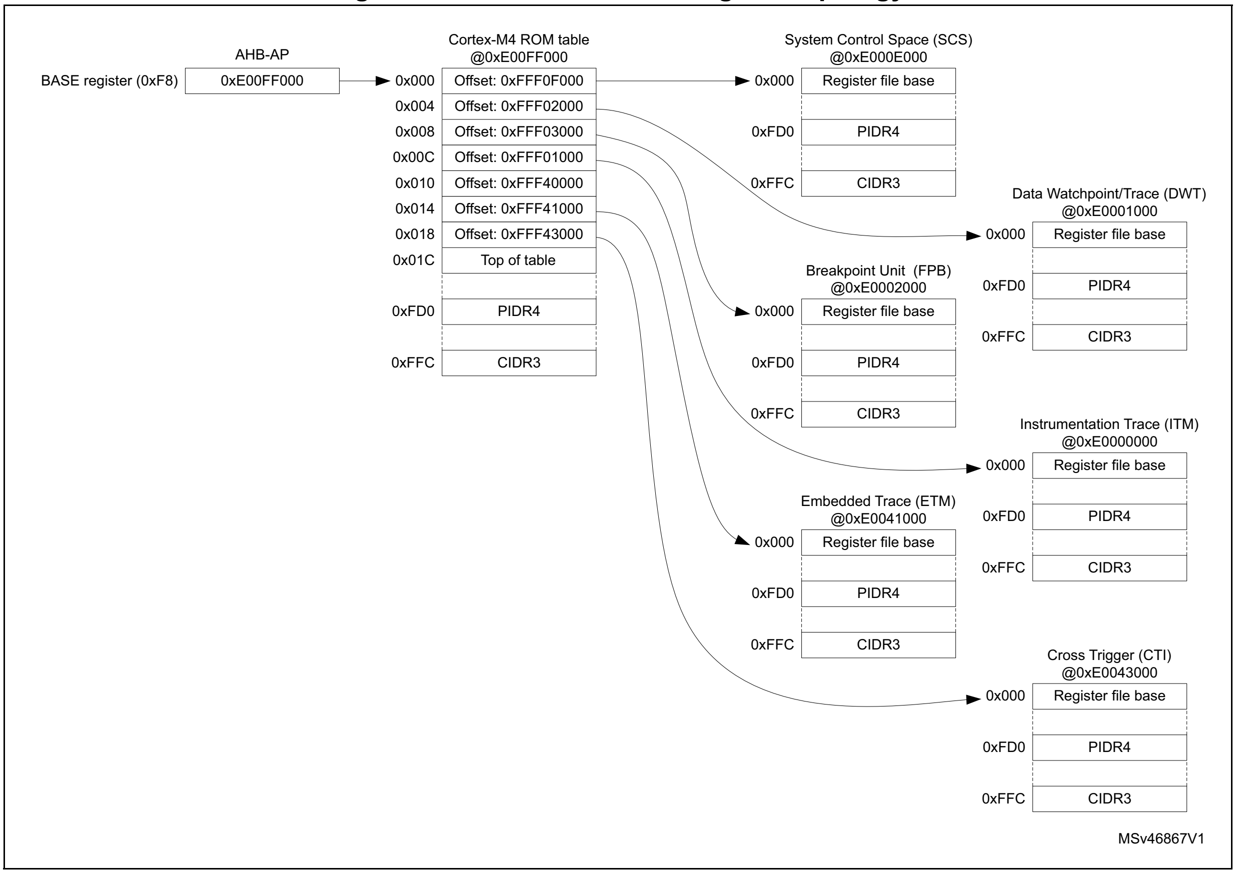
Table 593. Cortex®-M4 CPU ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4 Kbyte	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4 Kbyte	0xFFF0 2003
0xE00F F008	FPB	0xE000 2000	0xFFF0 3000	4 Kbyte	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4 Kbyte	0xFFF01 003
0xE00F F010	TPIU(1)	0xE004 0000	0xFFF4 1000	4 Kbyte	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4 Kbyte	0xFFF4 2003
0xE00F E018	CTI	0xE004 3000	0xFFF4 5000	4 Kbyte	0xFFF4 5003
0xE00F F01C	Top of table	-	-	-	0x0000 0000
0xE00F F020 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 594*

1. The TPIU is included in this table by default, but bit 0 is reset to indicate that it is not present.

The topology for the CoreSight™ components in the Cortex®-M4 subsystem is shown in *Figure 794*.

Figure 794. Cortex®-M4 CoreSight™ topology



Cortex®-M4 ROM registers

CM4ROM memory type register (CM4ROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x1: system memory is present on this bus

CM4ROM CoreSight peripheral identity register 4 (CM4ROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: No continuation code

CM4ROM CoreSight peripheral identity register 0 (CM4ROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x50: ROM table part number bits [7:0]

CM4ROM CoreSight peripheral identity register 1 (CM4ROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics identity code bits [3:0]

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: ROM table part number bits [11:8]

CM4ROM CoreSight peripheral identity register 2 (CM4ROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 JEP106ID[6:4]: JEP106 identity code bits [6:4]
0x2: STMicroelectronics identity code bits [6:4]

CM4ROM CoreSight peripheral identity register 3 (CM4ROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

CM4ROM CoreSight component identity register 0 (CM4ROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

CM4ROM CoreSight peripheral identity register 1 (CM4ROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

CM4ROM CoreSight component identity register 2 (CM4ROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: Common ID value

CM4ROM CoreSight component identity register 3 (CM4ROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Cortex®-M4 ROM registers map

Table 594. Cortex®-M4 CPU ROM table register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	**CM4ROM_ MEMTYPE**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	**CM4ROM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4	**CM4ROM_PIDR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	**CM4ROM_PIDR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	**CM4ROM_PIDR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	**CM4ROM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	0	1	0	0	0	0

Table 594. Cortex®-M4 CPU ROM table register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	**CM4ROM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	**CM4ROM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	**CM4ROM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**CM4ROM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**CM4ROM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	**CM4ROM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	**CM4ROM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.12.2 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:

- Watchpoint
- ETM trigger
- PC sampling trigger
- Data address sampling trigger
- Data comparator (comparator 1 only)
- Clock cycle counter comparator (comparator 0 only)

It also contains counters for:
- Clock cycles
- Folded instructions
- Load Store Unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares one of the following with the value held in its DWT_COMP register:

a) a data address
b) an instruction address
c) a data value
d) the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT Data trace packets, containing one or more of:
  a) the address of the instruction that caused a data access
  b) an address offset, bits[15:0] of the data access address
  c) the matched data value
- A watchpoint debug event, on either the PC value or the accessed data address
- A CMPMATCH[N] event, that signals the match outside the DWT unit

A watchpoint debug event either generates a Debug Monitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the *Arm®v7-M Architecture Reference Manual* [5].

Data watchpoint and trace unit registers

The Cortex®-M4 DWT registers are located at address range 0xE0001000 to 0xE0001FFC.

DWT control register (DWT_CTRL)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: number of comparators implemented
0x4: four comparators

Bit 27 **NOTRCPKT**: trace sampling and exception tracing support
0x0: supported

Bit 26 **NOEXTTRIG**: external match signal, CMPMATCH support
0x0: supported

Bit 25 **NOCYCCNT**: cycle counter support
0x0: supported

Bit 24 **NOPRFCNT**: profiling counter support
0x0: supported

Bit 23 Reserved, must be kept at reset value.

Bit 22 **CYCEVTENA**: enable for POSTCNT underflow event counter packet generation
0x0: disabled
0x1: enabled

Bit 21 **FOLDEVTENA**: enable for folded instruction counter overflow event generation
0x0: disabled
0x1: enabled

Bit 20 **LSUEVTENA**: enable for LSU counter overflow event generation
0x0: disabled
0x1: enabled

Bit 19 **SLEEPEVTENA**: enable for Sleep mode counter overflow event generation
0x0: disabled
0x1: enabled

Bit 18 **EXCEVTENA**: enable for exception overhead counter overflow event generation
0x0: disabled
0x1: enabled

Bit 17 **CPIEVTENA**: enable for CPI counter overflow event generation
0x0: disabled
0x1: enabled

Bit 16 **EXCTRCENA**: enable for exception trace generation
0x0: disabled
0x1: enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: PC sample enable
Enables the use of the POSTCNT counter as a timer for periodic PC sample packet generation.
0x0: disabled
0x1: enabled

Bits 11:10 **SYNCTAP[1:0]**: synchronization tap position
Selects the position of the synchronization packet counter tap on the CYCCNT counter. This determines the synchronization packet rate.
0x0: disabled. No synchronization packets
0x1: tap at CYCCNT[24]
0x2: tap at CYCCNT[26]
0x3: tap at CYCCNT[28]

Bit 9 **CYCTAP**: CYCCNT tap position
Selects the position of the POSTCNT tap on the CYCCNT counter.
0x0: tap at CYCCNT[6]
0x1: tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: POSTCNT counter initial value
Writes to this field are ignored if POSTCNT counter is enabled (meaning that CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTPRESET[3:0]**: POSTCNT counter reload value

Bit 0 **CYCCNTENA**: CYCCNT counter enable
0x0: disabled
0x1: enabled

DWT cycle count register (DWT_CYCCNT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CYCCNT[31:0]**: processor clock cycle counter

DWT CPI count register (DWT_CPICNT)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter
Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: exception overhead cycle counter
Counts the number of cycles spent in exception processing.

DWT Sleep mode count register (DWT_SLPCNT)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: sleep mode cycle counter
Counts the number of cycles spent in Sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNT)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: load store counter
Counts the additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNT)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]**: executed instruction address sample value
Samples the current value of the program counter.

DWT comparator x registers (DWT_COMPx)

Address offset: 0x020 + 0x010 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COMP[31:0]**: reference value for comparison

DWT comparator mask x registers (DWT_MASKx)

Address offset: 0x024 + 0x010 * x,(x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **MASK[4:0]**: comparator mask size

Provides the size of the ignore mask applied to the access address for address range matching by comparator x. A debugger can write 11111 to this field and then read the register back to determine the maximum mask size supported.

DWT comparator function x registers (DWT_FUNCTx)

Address offset: 0x028 + 0x010 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]			
							r					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
rw	rw	rw	rw	rw	rw	r	rw	rw		rw		rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: match occurred

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **DATAVADDR1[3:0]**: second comparator number
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12 **DATAVADDR0[3:0]**: comparator number
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10 **DATAVSIZE[1:0]**: data comparison size
For data value matching, specifies the size of the required data comparison.
0x0: byte
0x1: half word
0x2: word
0x3: reserved

Bit 9 **LNK1ENA**: second linked comparator support
Indicates whether use of a second linked comparator is supported.
0x1: supported

Bit 8 **DATAVMATCH**: cycle comparison enable
0x0: performs address comparison
0x1: performs data value comparison

Bit 7 **CYCMATCH**: cycle count comparison enable
Enables cycle count comparison on comparator 0. This field is reserved for other comparators.
0x0: no cycle count comparison
0x1: compares DWT_COMP0 with the cycle counter, DWT_CYCCNT

Bit 6 Reserved, must be kept at reset value.

Bit 5 **EMITRANGE**: address offset data trace enable
Enables generation of data trace address offset packets (containing data address bits 0 to 15).
0x0: disabled
0x1: enabled

Bit 4 Reserved, must be kept at reset value.

Bits 3:0 **FUNCTION[3:0]**: comparator match action
Selects the action to take on comparator match. The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See *[5]*.

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x02: DWT part number

DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x0: DWT part number

DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x3: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0xE: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0x05: common ID value

DWT register map

Table 595. DWT register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DWT_CTRL**	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	Res.	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP		CYCTAP	POSINIT[3:0]				POSTPRESET				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**DWT_CYCCNT**	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**DWT_CPICNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x00C	**DWT_EXCCNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x010	**DWT_SLPCNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	**DWT_LSUCNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	**DWT_FOLDCNT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	**DWT_PCSR**	EIASAMPLE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**DWT_COMP0**	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 595. DWT register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	DWT_MASK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x028	DWT_FUNCT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x02C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x030	DWT_COMP1	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DWT_MASK1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x038	DWT_FUNCT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x03C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x040	DWT_COMP2	COMP[31:0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	DWT_MASK2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0
0x048	DWT_FUNCT2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x050	DWT_COMP3	COMP[31:0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	DWT_MASK3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASK[4:0]				
	Reset value																												0	0	0	0	0

Table 595. DWT register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058	DWT_FUNCT3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	DATAVADDR1[3:0]				DATAVADDR0[3:0]				DATAVSIZE[1:0]		LNK1ENA	DATAVMATCH	CYCMATCH	Res.	EMITRANGE	Res.	FUNCTION[3:0]			
	Reset value								0					0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0
0x05C to 0xFCC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	DWT_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	DWT_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	DWT_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	1	0
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	1	1	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0

Table 595. DWT register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.12.3 Instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. Software trace.

   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access, and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.
2. Hardware trace.

   The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
3. Local timestamping.

   The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.
4. Global system timestamping.

   Timestamps can also be generated using the system-wide 64-bit count value coming from the Timestamp Generator component.

Instrumentation trace macrocell registers

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC.

ITM stimulus x registers (ITM_STIMx)

Address offset: 0x000 + 0x004 *x,(x = 0 to 31)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMULUS[31:0]**: stimulus port indicator

Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator.

0: stimulus port buffer is full (or port is disabled)

1: stimulus port can accept new write data

ITM trace enable register (ITM_TER)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: stimulus port enable

Each bit x (0:32) enables the stimulus port associated with the ITM_STIMx register.

0: port disabled

1: port enabled

ITM trace privilege register (ITM_TPR)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVMASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIVMASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVMASK[31:0]**: unprivileged access enable

Enables unprivileged access to ITM stimulus ports. Each bit controls eight stimulus ports.
XXX0: unprivileged access permitted on ports 0 to 7
XXX1: only privileged access permitted on ports 0 to 7
XX0X: unprivileged access permitted on ports 8 to 15
XX1X: only privileged access permitted on ports 8 to 15
X0XX: unprivileged access permitted on ports 16 to 23
X1XX: only privileged access permitted on ports 16 to 23
0XXX: unprivileged access permitted on ports 24 to 31
1XXX: only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	GTSFREQ[1:0]		TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
				rw	rw	rw	rw				r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: ITM busy

Indicates whether the ITM is currently processing events.
0: not busy
1: busy

Bits 22:16 **TRACEBUSID[6:0]:** identifier for multi-source trace stream formatting

If multi-source trace is in use, the debugger must write a non-zero value to this field. Different IDs must be used for each trace source in the system.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:10 **GTSFREQ[1:0]:** global timestamp frequency

Defines how often the ITM generates a global timestamp, based on the global timestamp clock frequency, or disables generation of global timestamps.
The possible values are:
0x0: disable generation of global timestamps
0x1: generate timestamp request whenever the ITM detects a change in global timestamp counter bits [47:7]. This is approximately every 128 cycles.
0x2: generate timestamp request whenever the ITM detects a change in global timestamp counter bits [47:13]. This is approximately every 8192 cycles.
0x3: generate a timestamp after every packet, if the output FIFO is empty.

Bits 9:8 **TSPRESCALE[1:0]**: local timestamp prescaler
This field is used with the trace packet reference clock.
The possible values are:
0x0: no prescaling
0x1: divide by 4
0x2: divide by 16
0x3: divide by 64

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SWOENA**: Enables asynchronous clocking of the timestamp counter
0: timestamp counter uses processor clock

Bit 3 **TXENA**: Enables forwarding of hardware event packets from the DWT unit to the trace port
0: disabled
1: enabled

Bit 2 **SYNCENA**: synchronisation packet transmission enable
If a debugger sets this bit it must also configure the DWT_CTRL register SYNCTAP field in the DWT for the correct synchronisation speed.
0: disabled
1: enabled

Bit 1 **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0 **ITMENA**: ITM enable
0: disabled
1: enabled

ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x01: ITM part number

ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x0: ITM part number

ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: Common ID value

ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0xE: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Table 596. ITM register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ITM_STIM0	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x004	ITM_STIM1	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x008	ITM_STIM2	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x00C	ITM_STIM3	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x010	ITM_STIM4	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x014	ITM_STIM5	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x018	ITM_STIM6	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x01C	ITM_STIM7	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x020	ITM_STIM8	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x024	ITM_STIM9	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x028	ITM_STIM10	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x02C	ITM_STIM11	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x030	ITM_STIM12	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x034	ITM_STIM13	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x038	ITM_STIM14	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x03C	ITM_STIM15	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x040	ITM_STIM16	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x044	ITM_STIM17	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x048	ITM_STIM18	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 596. ITM register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04C	ITM_STIM19	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x050	ITM_STIM20	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x054	ITM_STIM21	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x058	ITM_STIM22	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x05C	ITM_STIM23	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x060	ITM_STIM24	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x064	ITM_STIM25	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x068	ITM_STIM26	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x06C	ITM_STIM27	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x070	ITM_STIM28	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x074	ITM_STIM29	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x078	ITM_STIM30	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x07C	ITM_STIM31	STIMULUS[31:0]																															
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0x080 to 0xDFC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0E00	ITM_TER	STIMENA[31:0]																															
	Reset value																																
0xE04 to 0xE3C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0x0E40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
	Reset value																													0	0	0	0

Table 596. ITM register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE44 to 0xE7C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xE80	**ITM_TCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	GTSFREQ[1:0]		TSPRESCALE[1:0]		Res.	Res.	Res.	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0					0	0	0	0				0	0	0	0	0
0xE84 to 0xFCC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD0	**ITM_PIDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	**ITM_PIDR5**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	**ITM_PIDR6**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	**ITM_PIDR7**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	**ITM_PIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	**ITM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	**ITM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	**ITM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**ITM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1

Table 596. ITM register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.12.4 Breakpoint unit (FPB)

The FPB allows hardware breakpoints to be set. It contains six comparators which monitor the instruction fetch address and two literal address comparators. If a match occurs, the address is remapped to an address in system memory, defined by the FPB_REMAP register plus an offset corresponding to the matching comparator. Alternatively, the instruction comparators can be configured to generate a breakpoint instruction.

Breakpoint unit registers

The Cortex®-M4 FPB registers are located at address range 0xE000 2000 to 0xE000 2FFC.

FPB control register (FPB_CTRL)

Address offset: 0x000

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 11:8 **NUM_LIT[3:0]**: number of literal address comparators supported
0x2: two literal comparators supported.

Bits 14, 13, 12, 7, 6, 5, 4 **NUM_CODE[6:0]**: number of instruction address comparators supported
0x6: 6 instruction comparators supported

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: write protect key
A write to FPB_CTRL register will be ignored if this bit is not set to 1.

Bit 0 **ENABLE**: FPB enable
0x0: disable
0x1: enable

FPB remap register (FPB_REMAP)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	RMPSPT	REMAP[23:11]												
		r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REMAP[10:0]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **RMPSPT**: flash patch remap support
Indicates whether flash patch remap is supported.
0x1: remapping supported.

Bits 28:5 **REMAP**[23:0]: remap target address
Bits [28:5] of the base address in SRAM to which the FPB remaps the address. The remap base address must be aligned to the number of words required to support the implemented comparators, that is to (NUM_CODE + NUM_LIT) words, with a minimum alignment of 8 words. Because remap is into the SRAM memory region, 0x2000 0000-0x3FFF FFFF, bits [31:29] of the remap address are 001.

Bits 4:0 Reserved, must be kept at reset value.

FPB comparator x registers (FPB_COMPx)

Address offset: 0x008 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REPLACE[1:0]		Res.	COMP[26:14]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
COMP[13:0]														Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:30 **REPLACE[1:0]**: behavior at comparison match
Defines the behavior when a match occurs between the COMP field and the instruction fetch/literal address(1).
0x0: remap to address defined in FPB_REMAP register, plus 4n, where n is the comparator number.
0x1: breakpoint on lower half-word, upper half-word is unaffected
0x2: breakpoint on upper half-word, lower half-word is unaffected
0x3: breakpoint on both upper and lower half-words

Bit 29 Reserved, must be kept at reset value.

Bits 28:2 **COMP[26:0]**: value to compare with address bits 28:2 of accesses to instruction code memory (0x0000 0000 to 0x1FFF FFFF). If a match occurs, the action to be taken is defined by the REPLACE field.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: comparator enable
The comparator is only enabled if both this bit and the FPB ENABLE bit in the FPB_CTRL register are set.
0: disabled
1: enabled

1. FPB_COMP(0:5) correspond to the instruction address comparators. FPB_COMP(6:7) correspond to the literal address comparators.

FPB CoreSight peripheral identity register 4 (FPB_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: register file occupies a single 4-Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

FPB CoreSight peripheral identity register 0 (FPB_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x03: FPB part number

FPB CoreSight peripheral identity register 1 (FPB_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x0: FPB part number

FPB CoreSight peripheral identity register 2 (FPB_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p4

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

FPB CoreSight peripheral identity register 3 (FPB_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

FPB CoreSight component identity register 0 (FPB_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: Common ID value

FPB CoreSight peripheral identity register 1 (FPB_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0xE: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component ID bits [11:8]
0x0: common ID value

FPB CoreSight component identity register 2 (FPB_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

FPB CoreSight component identity register 3 (FPB_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component ID bits [31:24]
0xB1: common ID value

Table 597. FPB register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FPB_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	Reset value																		0	0	0	0	0	1	0	0	1	1	0			0	0
0x004	FPB_REMAP	Res.	Res.	RMPSP	REMAP[23:0]																								Res.	Res.	Res.	Res.	Res.
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x008	FPB_COMP0	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x00C	FPB_COMP1	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x010	FPB_COMP2	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x014	FPB_COMP3	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x018	FPB_COMP4	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x01C	FPB_COMP5	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0

Table 597. FPB register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	FPB_COMP6	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x024	FPB_COMP7	REPLACE[1:0]		Res.	COMP[26:0]																											Res.	ENABLE
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x028 to 0xFCC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD0	FPB_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4	FPB_PIDR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFD8	FPB_PIDR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFDC	FPB_PIDR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																																
0xFE0	FPB_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	1	1
0xFE4	FPB_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	FPB_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	FPB_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 597. FPB register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	FPB_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	FPB_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	FPB_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	FPB_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.12.5 Embedded trace macrocell (ETM)

The Cortex®-M4 ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex®-M4 core to be traced. In STM32MP153, the ETM is configured for instruction trace only, meaning that data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The addresses of memory locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the *Arm® CoreSight™ ETM™-Cortex®-M4 technical reference manual [7]*.

Embedded trace macrocell registers

The ETM registers are accessed by the debugger via the Cortex®-M4 AHB-AP, at address range 0xE004 1000 to 0xE004 1FFC.

Cortex-M4 ETM registers

ETM control register (M4_ETM_CR)

Address offset: 0x000

Reset value: 0x0000 0411

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	ETMEN	PROG	DBRQ	BO	STALL	Res.	Res.	Res.	Res.	Res.	Res.	PDN
				rw	rw	rw	rw	rw							rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TSEN**: Timestamp enable
0: Timestamping disabled
1: Timestamping enabled

Bits 26:12 Reserved, must be kept at reset value.

Bit 11 **ETMEN**: ETM enable
0: Trace output disabled
1: Trace output enabled

Bit 10 **PROG**: ETM programming
This bit must be set to 1 before programming the ETM. Tracing is prevented while this bit is set to 1.
0: ETM operational
1: ETM in programming state

Bit 9 **DBRQ**: Debug request.
When set to 1 and the trigger event occurs, the ETMDBGRQ output is asserted until HALTED is observed. This enables the Arm® processor to be forced into Debug state.

Bit 8 **BO**: Branch output
When set to 1 all branch addresses are output, even if the branch was because of a direct branch instruction. Setting this bit enables reconstruction of the program flow without having access to the memory image of the code being executed.
When this bit is set to 1, more trace data is generated, and this may affect the performance of the trace system. Information about the execution of a branch is traced regardless of the state of this bit.

Bit 7 **STALL**: Stall processor
The FIFOFULL output can be used to stall the processor to prevent overflow. The FIFOFULL output is only enabled when the stall processor bit is set to 1. When the bit is 0 the FIFOFULL output remains low at all times and the FIFO overflows if there are too many trace packets. Trace resumes without corruption once the FIFO has drained, if overflow does occur.

Bits 6:1 Reserved, must be kept at reset value.

Bit 0 **PDN**: ETM power down
This bit can be used by an implementation to control if the ETM is in a low power state. This bit must be cleared by the trace software tools at the beginning of a debug session.
When this bit is set to 1, writes to some registers and fields might be ignored. The following registers and fields are still write-accessible:
– ETMCR bit [0]
– ETMLAR
– ETMCLAIMSET register
– ETMCLAIMCLR register
When the ETMCR is written with this bit set to 1, bits other than bit [0] might be ignored.

ETM configuration code register (M4_ETM_CCR)

Address offset: 0x004

Reset value: 0x8C80 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IDPRES	Res.	Res.	Res.	CPMAC	TSPRES	CIDCMP[1:0]		FFPRES	NEXTO[2:0]			NEXTI[2:0]			SEQPRES
r				r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NCNT[2:0]			NMMDEC[4:0]					NDVCMP[3:0]				NADCMP[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **IDPRES**: ETM ID register present
1: ETMIDR register is present and defines the ETM architecture version in use

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **CPMAC**: Co-processor and memory access
1: Memory-mapped access to registers is supported

Bit 26 **TSPRES**: Trace start/stop block present
1: Trace start/stop block is present

Bits 25:24 **CIDCMP[1:0]**: Number of context ID comparators
0x0: Context ID comparators are not implemented.

Bit 23 **FFPRES**: FIFOFULL logic present
To use FIFOFULL the system must also support the function, as indicated by bit[8] of ETMSCR.

1: FIFOFULL logic is present in the ETM

Bits 22:20 **NEXTO[2:0]**: Number of external outputs
0x0: No external outputs are supported

Bits 19:17 **NEXTI[2:0]**: Number of external inputs
0x2: Two external inputs are supported

Bit 16 **SEQPRES**: Sequencer present
0: The sequencer is not present in the ETM.

Bits 15:13 **NCNT[2:0]**: Number of counters
0x1: One counter is implemented

Bits 12:8 **NMMDEC[4:0]**: Number of memory map decoders
0x0: No memory map decoder inputs are implemented.

Bits 7:4 **NDVCMP[3:0]**: Number of data value comparators
0x0: No data value comparators are implemented.

Bits 3:0 **NADCMP[3:0]**: Number of address comparator pairs
0x0: No address comparator pairs are implemented.

ETM trigger register (M4_ETM_TRIGGER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
0x21: Embedded-ICE watchpoint comparators 2 (from DWT)
0x22: Embedded-ICE watchpoint comparators 3 (from DWT)
0x23: Embedded-ICE watchpoint comparators 4 (from DWT)
0x40: Counter 1 at zero
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x6F: Hard-wired input, always true
Others: Reserved

ETM status register (M4_ETM_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGFL	TSSRSTAT	PROGVAL	UOVFL
												rw	rw	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **TRIGFL**: Trigger flag

Set when the trigger occurs, and prevents the trigger from being output until the ETM is programmed again.

Bit 2 **TSSRSTAT**: Trigger start/stop resource status

Holds the current status of the trace start/stop resource. If set to 1, it indicates that a trace on address has been matched, without a corresponding trace off address match.

Bit 1 **PROGVAL**: Programming bit value (read-only)

The current effective value of the ETM Programming bit, bit [10] of the M4_ETM_CR. Set the programming bit to 1 and wait for this bit to go to 1 before programming the ETM.
This bit remains at 0 until the FIFO is empty, or if OS lock is set in the M4_ETM_OSLSR register.

Bit 0 **UOVFL**: Untraced overflow (read-only)

If set to 1, there is an overflow that has not yet been traced. This bit is cleared to 0 when trace is restarted, or when the ETM power-down bit (bit [0] of the M4_ETM_CR register) is set to 1.

ETM configuration register (M4_ETM_SCR)

Address offset: 0x014

Reset value: 0x0002 0D09

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOFTCHCOMP	Res.
														r	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	NSUPPROC[2:0]			PORTMODESUP	PORTSIZESUP	MAXPORTSIZE[3]	FFSUP	Res.	Res.	Res.	Res.	Res.	MAXPORTSIZE[2:0]		
	r	r	r	r	r	r	r						r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **NOFTCHCOMP**: No fetch comparisons

1: Fetch comparisons are not implemented

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:12 **NSUPPROC[2:0]**: Number of supported processors

0x0: One processor supported

Bit 11 **PORTMODESUP**: Port mode support
0: Current selected port mode is not supported
1: Current selected port mode is supported

Bit 10 **PORTSIZESUP**: Port size support
0: Current selected port size is not supported
1: Current selected port size is supported

Bit 8 **FFSUP**: FIFOFULL support
1: FIFOFULL is supported

Bits 7:3 Reserved, must be kept at reset value.

Bits 9, 2, 1, 0 **MAXPORTSIZE[3:0]**: Maximum ETM port size field
These bits indicate the maximum ETM port size supported.
0x1: Maximum port size = 1

ETM trace enable event register (M4_ETM_TEEVR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function
If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier
See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier
Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
0x21: Embedded-ICE watchpoint comparators 2 (from DWT)
0x22: Embedded-ICE watchpoint comparators 3 (from DWT)
0x23: Embedded-ICE watchpoint comparators 4 (from DWT)
0x40: Counter 1 at zero
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x6F: Hard-wired input, always true
Others: Reserved

ETM trace enable control 1 register (M4_ETM_TECR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	ENON OFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **ENONOFF**: Trace start/stop trace enable control
0: Trace start/stop block does not control trace enable
1: Trace enable is controlled by the trace start/stop block

Bits 24:0 Reserved, must be kept at reset value.

ETM FIFOFULL level register (M4_ETM_FFLR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LEVEL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LEVEL[7:0]**: Threshold for FIFOFULL signal
The number of bytes left in the FIFO, below which the FIFOFULL signal is asserted. For example, setting this value to 15 causes processor stalling, if enabled, when there are less than 15 free bytes in the FIFO.

ETM counter reload value 1 register (M4_ETM_CNTRLDVR1)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INICNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INICNT[15:0]**: Initial count
Specifies the counter reload value.

ETM synchronization frequency register (M4_ETM_SYNCFR)

Address offset: 0x1E0

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	FREQUENCY[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FREQUENCY[11:0]**: Distance of synchronization points
This value is the time in bytes between synchronization points in the trace, used by ETM (the tools can start decompressing only at synchronization points).

0x400: fixed synchronization packet generation every 1024 bytes of trace

ETM ID register (M4_ETM_IDR)

Address offset: 0x1E4

Reset value: 0x4114 F250

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	ABPE	SXSUPP	T32SUPP	Res.	LDPCF
r	r	r	r	r	r	r	r				r	r	r		r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PROCFAM[3:0]				ETMARCHMAJ[3:0]				ETMARCHMIN[3:0]				REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Implementer code
0x41: Arm®

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ABPE**: Alternative ranch packet encoding
1: ABPE is implemented.

Bit 19 **SXSUPP**: Security extensions support
0: ETM behaves as if the processor is in secure mode at all times

Bit 18 **T32SUPP**: 32-bit Thumb instruction tracing
1: 32-bit Thumb instructions are traced as single instructions

Bit 17 Reserved, must be kept at reset value.

Bit 16 **LDPCF**: Load PC first
0: Data tracing is not supported

Bits 15:12 **PROCFAM[3:0]**: Processor family
0xF: Processor family is not identified in this register

Bits 11:8 **ETMARCHMAJ[3:0]**: Major ETM architecture version
0x2: Version 3

Bits 7:4 **ETMARCHMIN[3:0]**: Minor ETM architecture version
0x5: Version 5

Bits 3:0 **REVISION[3:0]**: Implementation revision
0x0: Revision 0

ETM configuration code extension register (M4_ETM_CCER)

Address offset: 0x1E8

Reset value: 0x1854 1800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TSSIZE	TSENC	RFC	Res.	Res.	Res.	Res.	TSIMP	EIIMP	TSSUWP	NUMWPIN[3:0]			
		r	r	r					r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NUMIR[2:0]			SUPPDAC	RR	XXINBUS[7:0]								XXINSEL[2:0]		
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TSSIZE**: Timestamp size
0: 48 bits.

Bit 28 **TSENC**: Timestamp encoding
1: Timestamp is encoded as a natural binary number

Bit 27 **RFC**: Reduced function counter
1: Counter is a reduced function counter

Bits 26:23 Reserved, must be kept at reset value.

Bit 22 **TSIMP**: Timestamping implemented
1: Timestamping implemented

Bit 21 **EIIMP**: EmbeddedICE behavior control implemented
0: Not implemented

Bit 20 **TSSUWP**: Trace start/stop uses Embedded ICE watchpoint inputs
1: Yes

Bits 19:16 **NUMWPIN[3:0]**: Embedded ICE watchpoint inputs
Number of inputs coming from the DWT.

0x4: Four inputs

Bits 15:13 **NUMIR[2:0]**: Instrumentation resources
0x0: None

Bit 12 **SUPPDAC**: Data address comparison support
1: Not supported

Bit 11 **RR**: Readable registers
1: All registers are readable

Bits 10:3 **XXINBUS[7:0]**: Extended external input bus
0x0: Not implemented

Bits 2:0 **XXINSEL[2:0]**: Extended external input selectors
0x0: Not implemented

ETM trace enable start/stop embedded ICE control register (M4_ETM_TESSEICR)

Address offset: 0x1F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPRS[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STARTRS[3:0]			
												rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **STOPRS[3:0]**: Stop resource selection
Setting any of these bits to 1 selects the corresponding EmbeddedICE watchpoint input as a TraceEnable stop resource. Bit [16] corresponds to input 1, bit [17] corresponds to input 2, bit [18] corresponds to input 3, and bit [19] corresponds to input 4.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **STARTRS[3:0]**: Start resource selection
Setting any of these bits to 1 selects the corresponding EmbeddedICE watchpoint input as a TraceEnable start resource. Bit [0] corresponds to input 1, bit [1] corresponds to input 2, bit [2] corresponds to input 3, and bit [3] corresponds to input 4.

ETM timestamp event register (M4_ETM_TSEVR)

Address offset: 0x1F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2]
															rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCN[1:0]		RESOURCEB[6:0]							RESOURCEA[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:14 **FCN[2:0]**: Boolean function

If A is defined as the first resource match and B as the second match, an event is defined as a function of A and B.

0x0: A
0x1: NOT A
0x2: A AND B
0x3: NOT A AND B
0x4: NOT A AND NOT B
0x5: A OR B
0x6: NOT A OR B
0x7: NOT A OR NOT B

Bits 13:7 **RESOURCEB[6:0]**: Second resource identifier

See RESOURCEA[6:0] field for bit description.

Bits 6:0 **RESOURCEA[6:0]**: First resource identifier

Bits [6:4] defines the resource type and bits [3:0] the index. The supported resource identifiers are listed below. Programming any other values may result in unpredictable behavior.
0x20: Embedded-ICE watchpoint comparators 1 (from DWT)
0x21: Embedded-ICE watchpoint comparators 2 (from DWT)
0x22: Embedded-ICE watchpoint comparators 3 (from DWT)
0x23: Embedded-ICE watchpoint comparators 4 (from DWT)
0x40: Counter 1 at zero
0x5F: Trace start/stop resource
0x60: External inputs 1
0x61: External inputs 2
0x6F: Hard-wired input, always true
Others: Reserved

ETM trace ID register (M4_ETM_TRACEIDR)

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace ID to output onto the trace bus

This should be programmed with a unique value to differentiate it from other trace sources in the system.

ETM ID register 2 (M4_ETM_IDR2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPTO	RFETO
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **SWPTO**: SWP or SWPB instruction order
Identifies the order for a SWP or SWPB instruction.

0: The Load transfer is traced before the Store transfer

Bit 0 **RFETO**: RFE instruction order
Identifies the order for a RFE instruction.

0: The PC transfer is traced before the CPSR transfer

ETM device power-down status register 2 (M4_ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POWER
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **POWER**: ETM powered
1: ETM powered; trace registers can be accessed.

ETM claim tag set register (M4_ETM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3

Read:
0xF: Indicates there are four bits in claim tag

ETM claim tag clear register (M4_ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

ETM lock access register (M4_ETM_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETM register write access

Enables write access to some ETM registers by processor cores (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access

Other values: Disable write access

ETM lock status register (M4_ETM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK TYPE	LOCK GRANT	LOCK EXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the ETM_LAR register

0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns zero when read by an external debugger.

0: Write access is permitted

1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism

The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.

0: No lock control mechanism exists

1: Lock control mechanism is implemented

ETM authentication status register (M4_ETM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented

ETM CoreSight device identity register (M4_ETM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (M4_ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETM CoreSight peripheral identity register 0 (M4_ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0025

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x25: ETM part number

ETM CoreSight peripheral identity register 1 (M4_ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETM part number

ETM CoreSight peripheral identity register 2 (M4_ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (M4_ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (M4_ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETM CoreSight component identity register 1 (M4_ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETM CoreSight component identity register 2 (M4_ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETM CoreSight component identity register 3 (M4_ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Embedded trace macrocell registers map

Table 598. Embedded trace macrocell register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**M4_ETM_CR**	Res.	Res.	Res.	Res.	TSEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETMEN	PROG	DBRQ	BO	STALL	Res.	Res.	Res.	Res.	Res.	Res.	PDN
	Reset value					0																0	1	0	0	0							1
0x004	**M4_ETM_CCR**	IDPRES	Res.	Res.	Res.	CPMAC	TSPRES	CIDCMP [1:0]		FFPRES	NEXTO [2:0]			NEXTI [2:0]			SEQPRES	NCNT [2:0]			NMMDEC [4:0]					NDVCMP [3:0]				NADCMP [3:0]			
	Reset value	1				1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**M4_ETM_ TRIGGER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB [6:0]							RESOURCEA [6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**M4_ETM_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGFL	TSSRSTAT	PROGVAL	UOVFL
	Reset value																													0	0	0	0
0x014	**M4_ETM_SCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOFTCHCOMP	Res.	Res.	NSUPPROC [2:0]			PORTMODESUP	PORTSIZESUP	MAXPORTSIZE [3]	FFSUP	Res.	Res.	Res.	Res.	Res.	MAXPORTSIZE [2:0]		
	Reset value															1			0	0	0	1	1	0	1						0	0	1
0x020	**M4_ETM_TEEVR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN [2:0]			RESOURCEB [6:0]							RESOURCEA [6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**M4_ETM_TECR1**	Res.	Res.	Res.	Res.	Res.	Res.	ENONOFF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0																									
0x028	**M4_ETM_FFLR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LEVEL [7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x140	**M4_ETM_ CNTRLDVR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INICNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1E0	**M4_ETM_SYNCFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FREQUENCY[11:0]												
	Reset value																				0	1	0	0	0	0	0	0	0	0	0	0	0

Table 598. Embedded trace macrocell register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E4	M4_ETM_IDR	DESIGNER[7:0]								Res.	Res.	Res.	ABPE	SXSUPP	T32SUPP	Res.	LDPCF	PROCFAM [3:0]				ETMARCHMAJ [3:0]				ETMARCHMIN [3:0]				REVISION [3:0]			
	Reset value	0	1	0	0	0	0	0	1				1	0	1	0	0	1	1	1	1	0	0	1	0	0	1	0	1	0	0	0	0
0x1E8	M4_ETM_CCER	Res.	Res.	TSSIZE	TSENC	RFC	Res.	Res.	Res.	Res.	TSIMP	EIIMP	TSSUWP	NUMWPIN [3:0]				NUMIR [3:0]			SUPPDAC	RR	XXINBUS [7:0]								XXINSEL [2:0]		
	Reset value			0	1	1					1	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0x1F0	M4_ETM_ TESSEICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPRS [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STARTRS [3:0]			
	Reset value													0	0	0	0													0	0	0	0
0x1F8	M4_ETM_TSEVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCN[2:0]			RESOURCEB [6:0]							RESOURCEA [6:0]						
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x200	M4_ETM_ TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID [6:0]						
	Reset value																										0	0	0	0	0	0	0
0x208	M4_ETM_IDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWPTO	RFETO
	Reset value																															0	0
0x314	M4_ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POWER
	Reset value																																1
0xFA0	M4_ETM_ CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	M4_ETM_ CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFB0	M4_ETM_LAR	ACCESS_W[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xFB4	M4_ETM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	M4_ETM_ AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									0	0	0	0	1	0	1	0
0xFCC	M4_ETM_DEV TYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	M4_ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	M4_ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM [7:0]							
	Reset value																									0	0	1	0	0	1	0	1

Table 598. Embedded trace macrocell register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	**M4_ETM_PIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	**M4_ETM_PIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	**M4_ETM_PIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	**M4_ETM_CIDR0**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	**M4_ETM_CIDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS [3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	**M4_ETM_CIDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	1	0	0	1
0xFFC	**M4_ETM_CIDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

64.12.6 Cross trigger interface (CTI)

See *Section 64.10.3*.

64.13 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013.
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p1 Technical Reference Manual, Issue F, 26th Sept 2013.
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B, 10 Dec 2010
4. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
5. DDI 0403D (ID100710) - Arm® v7-M Architecture Reference Manual, Issue Derrata2010_Q3, November 2010
6. DDI 0468A (ID101712) - Arm® CoreSight™ ETM™-A7 r0p0, Issue A, 12 Sept 2011
7. DDI 0440C (ID070610) - Arm® CoreSight™ ETM™-M4 r0p1 Technical Reference Manual, Issue C, 29 June 2012
8. DDI 0528B (ID062514) - Arm® CoreSight™ STM-500 System Trace Macrocell r0p1 Technical Reference Manual, Issue B, 11 March 2014
9. DDI 0464F (ID051113) - Arm® Cortex®-A7 MPCore™ r0p5 Technical Reference Manual, Issue F, 11 April 2013
10. IHI 0022E (ID022613)- Arm AMBA AXI and ACE Protocol Specification, Issue E, 22 February 2013
11. Arm IHI 0033A - Arm AMBA 3 AHB-Lite Protocol v1.0, Issue A, 6 June 2006
12. Arm IHI 0024C (ID041610) - Arm AMBA Protocol v2.0, Issue C, 13 April 2010

65 Device electronic signature

The device electronic signature are either set in HW and stored in the OTP (visible in BSEC registers) which can be read using the debug interface or by the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match the characteristics of the component.

65.1 Unique device ID register (96 bits) (UID)

The unique device identifier is ideally suited:

- for use as serial number (USB string serial number, or other end applications)
- for use as part of the security keys, to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- during processes such as secure boot.

The 96-bit unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Base address: 0x5C00 5000 (BSEC base address on APB5)

Address offset: 0x234

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** 31:0 unique ID bits

Address offset: 0x238

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[63:32]:** 63:32 unique ID bits

Address offset: 0x23C

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** 95:64 unique ID bits

65.2 Device part number (RPN)

Base address: 0x5C00 5000 (BSEC base address on APB5)

Address offset: 0x204

Reset value: 0xXXXX 80XX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Options[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPN_coding[7:0]							
								r	r	r	r	r	r	r	r

Bits 7:0 **Options[15:0]:** Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RPN_coding[7:0]:**
0x24: STM32MP153Cx
0x25: STM32MP153Ax
0xA4: STM32MP153Fx
0xA5: STM32MP153Dx
Others: Reserved

65.3 Device version

Base address: 0x5008 1000 (DBGMCU base address on Debug_APB)

Address offset: 0x000

Reset value: 0xXXXX X500

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Silicon revision

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device ID
0x500: STM32MP15xx

65.4 Package data register (PKG)

Base address: 0x5C00 5000 (BSEC base address on APB5)

Address offset: 0x240

Reset value: 0xXXXX XXXX

Note: X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PKG[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r	r											
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:27 **PKG[2:0]**: Package type
- 000: Custom packages
- 001: TFBGA257
- 010: TFBGA361
- 011: LFBGA354
- 100: LFBGA448
- Others: Reserved

Bits 26:0 Reserved for other purposes (See *Section 3: Boot and security and OTP control (BSEC)*).

66 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

67 Revision history

Table 599. Document revision history

Date	Revision	Changes
08-Feb-2019	1	Initial release.
21-Feb-2019	2	Updated *Section 3: Boot and security and OTP control (BSEC)*.
30-Jul-2019	3	Updated *Section 2.1.2: Memory Map organization*. Updated *Section 2.2.3: Master ports description*. Updated *Section 3.3.3: OTP security modes*. Updated *Section 3.3.9: Debug control*. Updated *Table 22: DDRPHYC I/O list*. Updated *Section 8.5.1: Signal set 0: to monitor the DDR utilization*. Updated *Section 8.6.9: DDRPERFM event counter x register (DDRPERFM_CNTx)*. Updated *Figure 27: Power control block diagram*. Updated *Table 29: PWR pin overview*. Updated *Section 9.3.8: PWR VSW domain*. Updated *Figure 79: Peripheral clock distribution for QUADSPI and FMC*. Updated *Section 10.7.81: RCC stop request set register (RCC_MP_SREQSETR)*. Updated *Section 10.7.82: RCC stop request clear register (RCC_MP_SREQCLRR)*. Updated *Section 10.7.83: RCC global control register (RCC_MP_GCR)*. Updated *Section 10.7.92: RCC PWR_LP delay control register (RCC_PWRLPDLYCR)*. Updated *Section 10.7.115: RCC global reset control set register (RCC_MP_GRSTCSETR)*. Updated *Section 10.7.133: RCC APB5 peripheral enable for MPU set register (RCC_MP_APB5ENSETR)*. Updated *Section 10.7.135: RCC APB5 peripheral enable for MPU clear register (RCC_MP_APB5ENCLRR)*. Updated *Section 10.7.136: RCC AHB5 peripheral enable for MCU set register (RCC_MC_AHB5ENSETR)*. Updated *Section 10.7.138: RCC AHB5 peripheral enable for MCU clear register (RCC_MC_AHB5ENCLRR)*. Updated *Section 10.7.185: RCC AHB5 peripheral enable for MCU set register (RCC_MC_AHB5LPENSETR)*. Updated *Section 10.7.218: RCC IWDG clock freeze set register (RCC_MP_IWDGFZSETR)*. Updated *Section 10.7.219: RCC IWDG clock freeze clear register (RCC_MP_IWDGFZCLRR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
30-Jul-2019	3 (continued)	Updated *Section 13.3.16: TrustZone security (only for GPIOZ)*. Updated *Section 13.4.12: GPIO secure configuration register (GPIOZ_SECCFGR)*. Updated *Section 14.1: I/O compensation cell*. Updated *Section 17.5.7: MDMA channel x transfer configuration register (MDMA_CxTCR)*. Updated *Section 22.5.6: GICD interrupt set-enable register (GICD_ISENABLER0)*. Updated *Section 22.5.7: GICD interrupt set-enable register x (GICD_ISENABLERx)*. Updated *Section 22.6.14: GICC active priority register (GICC_APR0)*. Updated *Section 22.8.12: GICV VM active priority register (GICV_APR0)*. Updated *Section 22.8.13: GICV VM CPU interface identification register (GICV_IIDR)*. Updated *Section 23.6.19: EXTI external interrupt selection register 1 (EXTI_EXTICR1)*. Updated *Section 23.6.20: EXTI external interrupt selection register 2 (EXTI_EXTICR2)*. Updated *Section 23.6.21: EXTI external interrupt selection register 3 (EXTI_EXTICR3)*. Updated *Section 23.6.22: EXTI external interrupt selection register 4 (EXTI_EXTICR4)*. Updated *Section 25.10: FMC register map*. Updated *Section 26.5.6: QUADSPI communication configuration register (QUADSPI_CCR)*. Updated *Section 28.3: ADC implementation*. Updated *Table 186: ADC internal input/output signals*. Updated *Section 28.4.3: ADC clocks*. Updated *Section 28.4.11: Channel selection (SQRx, JSQRx)*. Updated *Table 192: TSAR timings depending on resolution*. Updated *Section 28.4.33: Temperature sensor*. Updated *Figure 244: Temperature sensor channel block diagram*. Updated *Section 28.6.5: ADC configuration register 2 (ADC_CFGR2)*. Updated *Section 28.6.7: ADC sample time register 2 (ADC_SMPR2)*. Updated *Section 28.6.17: ADC injected channel y offset register (ADC_OFRy)*. Updated *Section 28.6.18: ADC injected channel y data register (ADC_JDRy)*. Updated *Section 28.7.2: ADC common control register (ADC_CCR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
30-Jul-2019	3 (continued)	Updated *Section 28.7.3: ADC common regular data register for dual mode (ADC_CDR)*. Updated *Section 28.7.4: ADC common regular data register for 32-bit dual mode (ADC_CDR2)*. Updated *Section 30.3: DAC implementation*. Updated *Figure 252: Dual-channel DAC block diagram*. Updated *Section 30.4.2: DAC pins and internal signals*. Updated *Table 211: DAC internal input/output signals*. Added *Table 212: DAC interconnection*. Updated *Section 30.4.7: DAC trigger selection*. Updated *Section 30.4.11: DAC channel modes*. Updated *Figure 260: DAC sample and hold mode phase diagram*. Updated *Section 30.4.12: DAC channel buffer calibration*. Updated *Section 30.4.14: Dual DAC channel conversion modes (if dual channels are available)*. Updated *Section 30.5: DAC in low-power modes*. Updated *Section 30.6: DAC interrupts*. Updated *Table 216: DAC interrupts*. Updated *Section 30.7.1: DAC control register (DAC_CR)*. Updated *Section 32.2: DFSDM main features*. Updated *Section 33.1: Introduction*. Updated *Figure 269: DCMI block diagram*. Updated *Section 33.3.2: DCMI pins and internal signals*. Updated *Section 33.3.5: DCMI physical interface*. Updated *Table 239: DCMI interrupts*. Updated *Section 35.3.3: Random number generation*. Updated *Section 35.3.4: RNG initialization*. Updated *Table 249: RNG interrupt requests*. Updated *Section 35.7.3: RNG data register (RNG_DR)*. Updated entire *Section 37: Hash processor (HASH)*. Updated *Section 38.3.3: Repetition counter*. Updated *Section 38.3.4: External trigger input*. Updated *Section 38.3.5: Clock selection*. Updated *Section 38.3.16: Using the break function*. Updated *Section 38.2: TIM1/TIM8 main features*. Updated *Section 38.4.2: TIMx control register 2 (TIMx_CR2)(x = 1, 8)*. Updated *Section 39.3.3: Clock selection*. Updated *Section 39.3.4: Capture/Compare channels*. Updated *Section 39.4.2: TIMx control register 2 (TIMx_CR2)(x = 2 to 5)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
30-Jul-2019	3 (continued)	Updated *Figure 437: Output stage of capture/compare channel (channel 1)*. Updated *Section 40.4.2: TIM12 control register 2 (TIM12_CR2)*. Updated *Section 40.5.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14)*. Updated *Section 41.5: TIM15 registers*. Updated *Section 41.5.2: TIM15 control register 2 (TIM15_CR2)*. Updated *Section 41.5.5: TIM15 status register (TIM15_SR)*. Updated *Section 41.6: TIM16/TIM17 registers*. Updated *Section 41.6.2: TIMx control register 2 (TIMx_CR2)(x = 16 to 17)*. Updated *Section 41.6.6: TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 16 to 17)*. Updated *Section 41.6.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)*. Updated *Section 43.7: LPTIM registers*. Updated *Section 46.4.1: IWDG block diagram*. Updated *Section 46.4.3: Software and hardware watchdog modes*. Updated *Section 46.7.4: IWDG status register (IWDG_SR)*. Updated *Table 337: TAMP input/output pins*. Updated *Section 49.3.5: Tamper detection*. Updated *Section : Overrun error*. Updated *Section 51.8: USART registers*. Updated *Section 51.8.12: USART receive data register (USART_RDR)*. Updated *Section 51.8.13: USART transmit data register (USART_TDR)*. Updated *Section 52.11.1: SPI/I2S control register 1 (SPI_CR1)*. Updated *Section 52.11.2: SPI control register 2 (SPI_CR2)*. Updated *Section 52.11.3: SPI configuration register 1 (SPI_CFG1)*. Updated *Section 53.4.12: SPDIF output*. Updated *Table 392: SAI interrupt sources*. Updated *Section 53.6: SAI registers*. Updated *Section 57.4.1: Operating modes*. Updated *Section 57.4.3: Message RAM*. Updated *Section 57.4.7: Application*. Updated *Section 57.4.8: TTCAN operations (FDCAN1 only)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
30-Jul-2019	3 (continued)	Updated *Section 57.4.21: FDCAN Tx event FIFO element*. Updated *Section 57.4.22: FDCAN standard message ID filter element*. Updated *Section 57.4.23: FDCAN extended message ID filter element*. Updated *Section 57.5.6: FDCAN CC control register (FDCAN_CCCR)*. Updated *Section 57.5.15: FDCAN interrupt register (FDCAN_IR)*. Updated *Section 57.5.21: FDCAN extended ID filter configuration register (FDCAN_XIDFC)*. Updated *Section 57.6.6: FDCAN TT operation control register (FDCAN_TTOCN)*. Updated *Table 467: Device-mode control and status registers*. Updated *Section 58.14.2: OTG interrupt register (OTG_GOTGINT)*. Updated *Section 58.14.5: OTG reset register (OTG_GRSTCTL)*. Updated *Section 58.14.6: OTG core interrupt register (OTG_GINTSTS)*. Updated *Section 58.14.18: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*. Updated *Section 58.14.22: OTG host frame interval register (OTG_HFIR)*. Updated *Section 58.14.33: OTG host channel x transfer size register (OTG_HCTSIZx)*. Updated *Section 58.14.39: OTG device configuration register (OTG_DCFG)*. Updated *Section 58.14.40: OTG device control register (OTG_DCTL)*. Updated *Section 58.14.41: OTG device status register (OTG_DSTS)*. Updated *Section 58.14.50: OTG device IN endpoint x control register (OTG_DIEPCTLx)*. Updated *Section 58.14.51: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*. Updated *Section 58.15.3: Device initialization*. Updated *Section : Operational model*.
18-Feb-2020	4	**Document conventions:** Updated *Section 1.4: Glossary* **Memory and bus architecture:** Updated *Figure 1: Bus interconnect for STM32MP153*. Updated *Figure 4: Memory map*. Updated *Table 9: Register boundary addresses*. **BSEC:** Updated *Figure 5: BSEC block diagram*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	**OTP:** Updated *Table 17: OTP list words 0 to 31*. **DDRCTRL:** Updated *Section 5.4.2: Port arbiter (PA)*. Updated *Section 5.4.6: DFI Interface (DFI)*. Updated *Section 5.5.1: Transaction service control (TSC) and quality of service (QoS)*. Updated *Section 5.5.3: Power saving features*. Updated *Section 5.5.4: Address mapper*. **TZC:** Updated *Figure 6.6.16: TZC region x ID access register (TZC_REGION_ID_ACCESSx)*. **DDRPHY:** Updated *Section 7.1: DDRPHYC features*. Updated *Section 7.5: PUBL*. Updated *Section Figure 20.: PUBL block diagram*. Updated *Section 7.7.2: DDRPHYC PHY initialization register (DDRPHYC_PIR)*. Updated *Section 7.7.3: DDRPHYC PHY global control register (DDRPHYC_PGCR)*. Updated *Section 7.7.7: DDRPHYC PT register 0 (DDRPHYC_PTR0)*. Updated *Section 7.7.19: DDRPHYC MR1 register for LPDDR2 [alternate] (DDRPHYC_LPDDR2_MR1)*. Updated *Section 7.7.34: DDRPHYC ZQ0CR1 register (DDRPHYC_ZQ0CR1)*. Updated *Section 7.7.40: DDRPHYC byte lane n DLLC register (DDRPHYC_DXnDLLCR)*. Updated *Section 7.7.41: DDRPHYC byte lane n DQT register (DDRPHYC_DXnDQTR)*. Updated *Section 7.7.42: DDRPHYC byte lane n DQST register (DDRPHYC_DXnDQSTR)*. **RCC:** Updated *Section 10.3.6: Backup domain reset*. Updated *Figure 54: Top-level clock tree*. Updated *Section : CSS on LSE* Updated *Figure 60: LSE and HSE CSS function*. Updated *Figure 62: PLL3 and PLL4 block diagram*. Updated *Section 10.4.5: PLLs description*. Updated *Section : PLL programming recommendations*. Updated *Section : Using the PLLs in integer mode*. Updated *Section : Using the PLLs in fractional mode*. Updated *Section : Using the PLLs in spread spectrum mode*. Updated *Figure 66: Core and busses clock generation*. Updated *Table 56: Peripheral clock distribution overview* Updated *Section : Clock distribution for TIMs and LPTIMs*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	Updated *Figure 85: Kernel clock distribution for ADC12, DACs, STGEN and HDMI-CEC*. Updated *Figure 98: MPU low-power interface*. Updated *Figure 99: Interrupt block diagram*. Updated *Figure 102: MCU clock path*. Updated *Table 71: Clock settings examples for MPU* Updated *Section : PLL initialization procedure*. Updated *Section 10.6.3: Configuring the sub-system clocks*. Updated *Section 10.7.38: RCC PLL3 control register (RCC_PLL3CR)*. Removed Register 0x890. Updated *Section 10.7.42: RCC PLL4 control register (RCC_PLL4CR)*. Removed Register 0x8A4. Updated *Section 10.7.181: RCC APB5 sleep clock enable for MPU set register (RCC_MP_APB5LPENSETR)*. Updated *Section 10.7.183: RCC APB5 sleep clock enable for MPU clear register (RCC_MP_APB5LPENCLRR)*. **PWR:** Updated *Section 9.4: PWR power supply supervision* Updated *Section 9.4.6: PWR battery voltage thresholds* Updated *Table 33: System Low-power mode summary*. Updated *Table 34: Functionalities depending on system operating mode*. Updated *Table 46: PWR status flags*. Updated *Section 9.11.3: PWR control register 2 (PWR_CR2)* **GPIO:** Updated *Section 13.3.2: I/O pin alternate function multiplexer and mapping* Updated *Section 13.4.25: GPIO identification register (GPIOx_IPIDR) (x = A to K, Z)* **SYSCFG:** Updated *Figure 125: Ethernet clock source*. Updated *Section 14.3.4: SYSCFG IO control set register (SYSCFG_IOCTRLSETR)*. Updated *Section 14.3.5: SYSCFG IO control clear register (SYSCFG_IOCTRLCLRR)*. Updated *Section 14.3.9: SYSCFG compensation cell enable clear register (SYSCFG_CMPENCLRR)*. **ETZPC:** Update *Table 94: DECPROT assignment*. **MDMA:** Updated *Section 17.2: MDMA main features*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	**DMA:** Updated *Section 18.5.5: DMA stream x configuration register (DMA_SxCR)*. **GIC:** Updated *Section 22.5.4: GICD implementer identification register (GICD_IIDR)*. **FMC:** Added *Section 25.2: FMC implementation*. Updated *Figure 142: FMC block diagram*. Updated *Section 25.6: AXI interface*. Updated *Figure 144: FMC memory banks (Default mapping)*. Updated *Section 25.7.2: NAND flash memory address mapping*. Updated all table titles in *Section 25.8.4: NOR flash/PSRAM controller asynchronous transactions* and *Section 25.8.5: Synchronous transactions*. Updated *Section : SRAM/NOR flash chip-select control register for memory region x (FMC_BCRx)*. Updated *Section : NAND flash programmable control register (FMC_PCR)*. Added *Section : FMC NAND interrupt enable register (FMC_IER)*. Updated *Section : FMC hardware configuration register 1 (FMC_HWCFGR1)*. Updated *Section : FMC hardware configuration register 2 (FMC_HWCFGR2)*. **QUADSPI:** Updated *Section 26.3.4: QUADSPI signal interface protocol modes*. Updated *Section 26.3.11: QUADSPI configuration*. Updated *Section 26.3.16: NCS behavior*. Updated *Section 26.5.1: QUADSPI control register (QUADSPI_CR)*. Updated *Section 26.5.7: QUADSPI address register (QUADSPI_AR)*. **ADC:** Updated *Section 28.2: ADC main features*. Updated *Table 184: ADC features*. Updated *Table 185: ADC input/output pins*. Updated *Table 186: ADC internal input/output signals*. Updated *Section 28.4.3: ADC clocks*. Updated *Figure 178: ADC clock scheme*. Updated *Figure 180: ADC2 connectivity*. Updated *Section 28.4.6: ADC deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*. Updated *Section 28.4.8: Calibration (ADCAL, ADCALDIF, ADCALLIN, ADC_CALFACT)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	Updated *Figure 181: ADC calibration*. Updated *Section 28.4.11: Channel selection (SQRx, JSQRx)*. Updated *Table 190: ADC1, ADC2 - External triggers for regular channels*. Updated *Table 191: ADC1, ADC2- External triggers for injected channels*. Updated *Table 192: TSAR timings depending on resolution*. Updated *Section : ADC overrun (OVR, OVRMOD)*. Updated *Figure 210: Example of overrun (OVRMOD = 0)*. Added *Figure 211: Example of overrun (OVRMOD = 1)*. Updated *Section : Managing a sequence of conversion without using the DMA*. Added *Section 28.4.28: Managing conversions using the DFSDM*. Added *Section : DFSDM mode in dual ADC mode interleaved mode*. Updated *Section 28.4.33: Temperature sensor*. Updated *Section 28.4.34: VBAT supply monitoring*. Updated *Section 28.4.35: Monitoring the internal voltage reference*. Updated *Section 28.6.1: ADC interrupt and status register (ADC_ISR)*. Updated *Section 28.6.26: ADC calibration factors register (ADC_CALFACT)*. Updated *Section 28.7.2: ADC common control register (ADC_CCR)*. Added *Section 28.7.5: ADC hardware configuration register (ADC_HWCFGR0)*. Added *Section 28.7.6: ADC version register (ADC_VERR)*. Added *Section 28.7.7: ADC identification register (ADC_IPIDR)*. Added *Section 28.7.8: ADC size identification register (ADC_SIDR)*.Updated *Section 28.4.33: Temperature sensor*. Updated *Section 28.4.34: VBAT supply monitoring*. Updated *Section 28.4.35: Monitoring the internal voltage reference*. Updated *Section 28.6.26: ADC calibration factors register (ADC_CALFACT)*. Updated *Section 28.7.2: ADC common control register (ADC_CCR)*. Added *Section 28.7.5: ADC hardware configuration register (ADC_HWCFGR0)*. Added *Section 28.7.6: ADC version register (ADC_VERR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	Added *Section 28.7.7: ADC identification register (ADC_IPIDR)*. Added *Section 28.7.8: ADC size identification register (ADC_SIDR)*. **DTS:** Updated *Figure 248: Temperature sensor functional block diagram*. Updated *Table 205: Trigger configuration*. Updated *Section 29.3.11: On-off control and ready flag*. Updated *Section 29.6: DTS registers* introduction. Updated *Section 29.6.1: Temperature sensor configuration register 1 (DTS_CFGR1)*. Updated *Section 29.6.6: Temperature sensor status register (DTS_SR)*. **DFSDM:** Updated *Section : DFSDM clocks*. Updated *Section 32.7: DFSDM channel y registers (y=0..7)* introduction. Updated *Section 32.7.5: DFSDM channel y data input register (DFSDM_CHyDATINR)* introduction. Updated *Section 32.8: DFSDM filter x module registers (x=0..5)* introduction. **DCMI:** Updated *Section 33.3: DCMI functional description*. Updated *Table 229: DCMI input/output pins*. Updated *Figure 270: DCMI signal waveforms*. Updated *Figure 274: Coordinates and size of the window after cropping*. **RNG:** Updated *Section 35.3.3: Random number generation*. Updated *Figure 283: Entropy source model*. Updated *Figure 284: RNG initialization overview*. Updated *Section 35.3.5: RNG operation*. **CRYP:** Updated *Section 36.2: CRYP main features*. Updated *Section 36.4.5: CRYP procedure to perform a cipher operation*. Updated *Section 36.4.17: CRYP key registers*. Updated *Table 265: CRYP interrupt requests*. Updated *Section 28.6.26: ADC calibration factors register (ADC_CALFACT)*. Updated *Section 28.7.2: ADC common control register (ADC_CCR)*. Added *Section 28.7.5: ADC hardware configuration register (ADC_HWCFGR0)*. Added *Section 28.7.6: ADC version register (ADC_VERR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	Added *Section 28.7.7: ADC identification register (ADC_IPIDR)*. Added *Section 28.7.8: ADC size identification register (ADC_SIDR)*. **TIM15/16/17:** Updated *Section 41.5.20: TIM15 input selection register (TIM15_TISEL)*. Updated *Section 41.6.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)*. **LPTIM:** Updated *Section 43.4.5: Glitch filter*. **IWDG:** Updated *Section 46.4.3: Software and hardware watchdog modes*. Updated *Section 46.7.4: IWDG status register (IWDG_SR)*. **SPI:** Updated *Section 52.4.5: Standard multislave communication*. Updated *Figure 587: Waveform examples*. Updated *Figure 588: Master I2S Philips protocol waveforms (16/32-bit full accuracy)*. Updated *Figure 589: I2S Philips standard waveforms*. Updated *Figure 590: Master MSB Justified 16-bit or 32-bit full-accuracy length*. Updated *Figure 591: Master MSB justified 16 or 24-bit data length*. Updated *Figure 592: Slave MSB justified*. Updated *Figure 593: LSB justified 16 or 24-bit data length*. Updated *Figure 597: Startup sequence, I2S Philips standard, master*. Updated *Figure 599: Stop sequence, I2S Philips standard, master*. Updated *Section 52.11.1: SPI/I2S control register 1 (SPI_CR1)*. **SAI:** Updated *Figure 615: Start-up sequence*. **SPDIF:** Updated *Section 54.2: SPDIFRX main features*. Updated *Section 54.5.1: SPDIFRX control register (SPDIFRX_CR)* introduction. **SDMMC:** Added *Section 56.1: SDMMC main features*. **FDCAN:** Added *Section 57.3: FDCAN implementation*. Updated *Section : Rx handling*.

Table 599. Document revision history (continued)

Date	Revision	Changes
18-Feb-2020	4 (continued)	Updated *Section : Operating conditions*. Updated *Section 57.4.7: Application*. Updated *Section 57.5.31: FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)*. Updated *Section 57.7.2: Calibration configuration register (FDCAN_CCU_CCFG)*. **CEC:** Updated *Section 62.7.6: CEC interrupt enable register (CEC_IER)*. **ETH:** Added *Table 475: Ethernet peripheral pins*. Updated *Table 476: Ethernet internal input/output signals*. Updated *Figure 732: Ethernet high-level block diagram*. Updated *Section : PTP timestamp offload function*. Updated *Section : Channel x status register (ETH_DMACxSR)*. Updated *Section : Tx queue 0 quantum weight register (ETH_MTLTXQ0QWR)*. Updated *Section : Tx queue x quantum weight register (ETH_MTLTXQxQWR)*. Updated *Section : Extended operating mode configuration register (ETH_MACECR)*. Updated *Section : Rx queue control 0 register (ETH_MACRXQC0R)*. Updated *Section : HW feature 1 register (ETH_MACHWF1R)*. Updated *Section : ARP address register (ETH_MACARPAR)*. Updated *Section : L3 and L4 control 0 register (ETH_MACL3L4C0R)*. Updated *Section : L3 and L4 control 1 register (ETH_MACL3L4C1R)*. Updated *Section : Subsecond increment register (ETH_MACSSIR)*. **DBG:** Updated *Section : FPB CoreSight component identity register 0 (FPB_CIDR0)*. **Device electronic signature:** Updated *Section 65.2: Device part number (RPN)*. Updated *Section 65.4: Package data register (PKG)*.
15-Jan-2021	5	**BSEC:** Updated *Section 3.3.2: Interface to OTP*. Updated *Section 3.3.6: OTP operations*. Updated footnote in *Table 14: Write access permissions*. Updated footnote in *Table 15: Read access permissions*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	Updated *Section 3.6.5: BSEC OTP lock configuration register (BSEC_OTP_LOCK)*. Updated *Section 3.6.16: BSEC shadow register x (BSEC_OTP_DATAx)*. **OTP:** Updated *Table 17: OTP list words 0 to 31*. **DDRCTRL:** Updated *Figure 7: DDRCTRL block diagram*. Updated *Section 5.6: DDRCTRL configuration* introduction. Updated *Section 5.7.3: DDRCTRL mode register read/write control register 0 (DDRCTRL_MRCTRL0)*. Updated *Section 5.7.78: DDRCTRL port x read Q0S configuration register 1 (DDRCTRL_PCFGQOS1_x)*. **DDRPHY:** Updated *Figure 12: DDRPHYC block diagram*. Updated *Section 7.7.1: DDRPHYC revision ID register (DDRPHYC_RIDR)*. Updated *Section 7.7.3: DDRPHYC PHY global control register (DDRPHYC_PGCR)*. Updated *Section 7.7.5: DDRPHYC DDR DLL global control register (DDRPHYC_DLLGCR)*. Updated *Section 7.7.10: DDRPHYC ACIOC register (DDRPHYC_ACIOCR)*. Updated *Section 7.7.14: DDRPHYC DTP register 0 (DDRPHYC_DTPR0)*. Updated *Section 7.7.15: DDRPHYC DTP register 1 (DDRPHYC_DTPR1)*. Updated *Section 7.7.16: DDRPHYC DTP register 2 (DDRPHYC_DTPR2)*. Updated *Section 7.7.27: DDRPHYC ODTC register (DDRPHYC_ODTCR)*. Updated *Section 7.7.37: DDRPHYC byte lane n GC register (DDRPHYC_DXnGCR)*. **PWR:** Updated *Section 9.3.8: PWR VSW domain*. Updated *Figure 36: Power-on reset/power-down reset waveform*. Updated *Figure 37: BOR thresholds*. Added footnote to *Figure 34: Functionalities depending on system operating mode*. Updated *Section 9.6.9: PWR LP-Stop and LPLV-Stop modes*. **RCC:** Updated *Section : Clock distribution for TIMs and LPTIMs*. Moved *Section : Glitch skipper (GSKP)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	Updated *Section 10.7.25: RCC APB1 clock divider register (RCC_APB1DIVR)*. Updated *Section 10.7.26: RCC APB2 clock divider register (RCC_APB2DIVR)*. Updated *Section 10.7.27: RCC APB3 clock divider register (RCC_APB3DIVR)*. Updated *Section 10.7.67: RCC FDCAN kernel clock selection register (RCC_FDCANCKSELR)*. **HSEM:** Updated entire *Section 11: Hardware semaphore (HSEM)*. **IPCC:** Added *Table 79: IPCC interface signals*. Updated *Section 12.3.4: IPCC interrupts*. Updated *Section 12.4.4: IPCC processor 1 to processor 2 status register (IPCC_C1TOC2SR)*. Updated *Section 12.4.8: IPCC processor 2 to processor 1 status register (IPCC_C2TOC1SR)*. Updated *Section 12.4.9: IPCC hardware configuration register (IPCC_HWCFGR)*. **GPIO:** Updated *Section 13.3.2: I/O pin alternate function multiplexer and mapping*. **ETZPC:** Updated *Section 15.7: ETZPC registers* introduction. Block interconnect: Updated *Section 16.1.1: Introduction*. **MDMA:** Updated *Section 17.5.6: MDMA channel x control register (MDMA_CxCR)*. **DMA:** Updated *Figure 128: DMA block diagram* **DMAMUX:** Updated *Table 111: DMAMUX: assignment of multiplexer inputs to resources*. Updated *Section 19.4.4: DMAMUX request line multiplexer*. **Interrupt list:** Added note to *Table 119: STM32MP153 EXTI Events*. **GIC:** Updated *Section 22.5.21: GICD shared peripheral interrupt register x (GICD_SPISRx)*. **EXTI:** Updated *Section 23: Extended interrupt and event controller (EXTI)* introduction. Updated *Section 23.2.1: EXTI connections between peripherals and CPU*. Updated *Section 23.3: EXTI functional description*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	Updated footnote of *Table 127: Masking functionality*. Updated *Section 23.6.23: EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR1)* title. Supressed *Section 24.6.xx: EXTI CPU wakeup with event mask register (EXTI_C1EMR1)* title. Updated *Section 23.6.26: EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR2)* title. Supressed *Section 24.6.xx: EXTI CPU wakeup with event mask register (EXTI_C1EMR2)* title. Updated *Section 23.6.29: EXTI CPU wakeup with interrupt mask register (EXTI_C1IMR3)* title and reset value. Suppressed *Section 24.6.xx: EXTI CPU wakeup with event mask register (EXTI_C1EMR3)* title. Updated *Section 23.6.32: EXTI hardware configuration register x (EXTI_HWCFGRx)* reset value. **FMC:** Updated *Section 25.1: FMC main features*. Added note in *Section 25.7.2: NAND flash memory address mapping*. Updated *Section 25.8.4: NOR flash/PSRAM controller asynchronous transactions*. Updated *Table 158: FMC_BCRx bitfields (mode D)*. Updated *Table 161: FMC_BCRx bitfields (muxed mode)*. Updated *Section 25.8.6: NOR/PSRAM controller registers*. Added *Section : FMC size identification register (FMC_SIDR)*. Added *Section : FMC identification register (FMC_IPIDR)*. Added *Section : FMC version register (FMC_VERR)*. Added *Section : FMC hardware configuration register 1 (FMC_HWCFGR1)*. Added *Section : FMC hardware configuration register 2 (FMC_HWCFGR2)*. Updated *Section 25.9.5: NAND flash prewait function*. Updated *Figure 163: Access to NAND flash memory*. Updated *Section : BCH (Bose, Chaudhuri and Hocquenghem) code*. Updated *Section 25.9.7: FMC command sequencer*. Updated *Figure 165: Example of 2 Kbyte page program sequence with 4-bit BCH*. Updated *Section : Page read sequence*. Updated *Figure 166: Chained-page read command timings*. Updated *Figure 166: Chained-page read command timings*. **DLYB:** Updated *Section 27.2: DLYB main features*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	Updated *Section 27.3: DLYB functional description*. **DTS:** Updated *Section 29.6.2: Temperature sensor T0 value register 1 (DTS_T0VALR1)*. **DAC:** Updated *Section : Sample and hold mode*. Updated formula in *Section 30.4.12: DAC channel buffer calibration*. **VREFBUF:** Updated *Section 31.3.1: VREFBUF control and status register (VREFBUF_CSR)*. Updated *Section 31.3.2: VREFBUF calibration control register (VREFBUF_CCR)*. **LTDC:** deeply revisited section. **DSI:** Added*Section 36.4.2: DSI Host pins and internal signals*. Updated *Section : Synchronization with the LTDC*. Updated *Section 36.16.8: DSI Wrapper regulator and PLL control register (DSI_WRPCR)*. **HASH:** updated *Section 37.4.5: Message digest computing*. updated *Section 37.4.7: HMAC operation*. updated *Section 37.6.1: HASH control register (HASH_CR)*. **CRYP:** deeply revisited section. **TIM1/TIM8** Updated *Figure 339: Control circuit in external clock mode 2*. Updated *Figure 341: Capture/compare channel 1 main circuit*. Updated *Section 38.3.26: Timer synchronization*. Added Note in *Section 38.4.1: TIMx control register 1 (TIMx_CR1)(x = 1, 8)*. Updated *Section 38.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)*. Updated *Table 279: Output control bits for complementary OCx and OCxN channels with break feature*. Updated *Section 38.4.26: TIM1 alternate function option register 1 (TIM1_AF1)*. Updated *Section 38.4.27: TIM1 Alternate function register 2 (TIM1_AF2)*. Updated *Section 38.4.28: TIM8 Alternate function option register 1 (TIM8_AF1)*. Updated *Section 38.4.29: TIM8 Alternate function option register 2 (TIM8_AF2)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	**TIM2/TIM3/TIM4/TIM5:** Updated *Figure 372: General-purpose timer block diagram*. Updated *Figure 396: Control circuit in external clock mode 2*. Updated *Figure 398: Capture/Compare channel 1 main circuit*. Updated *Figure 399: Output stage of Capture/Compare channel (channel 1)*. Updated *Section 39.3.19: Timer synchronization*. Updated *Figure 416: Master/slave connection example with 1 channel only timers*. Updated *Section 39.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)*. Updated *Section 39.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)*. **TIM12/TIM13/TIM14:** Updated *Figure 423: General-purpose timer block diagram (TIM13/TIM14)*. Updated *Figure 436: Capture/compare channel 1 main circuit*. Updated *Figure 437: Output stage of capture/compare channel (channel 1)*. Added *Section 40.3.18: Using timer output as trigger for other timers (TIM13/TIM14)*. Updated *Section 40.4.8: TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1)*. Updated *Section 40.4.9: TIM12 capture/compare enable register (TIM12_CCER)*. Updated *Section 40.5.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14)*. Updated *Section 40.5.7: TIMx capture/compare enable register (TIMx_CCER)(x = 13 to 14)*. **TIM15/TIM16/TIM17:** Updated *Figure 449: TIM16/TIM17 block diagram*. Updated *Figure 463: Capture/compare channel 1 main circuit*. Updated *Figure 464: Output stage of capture/compare channel (channel 1)*. Updated *Figure 465: Output stage of capture/compare channel (channel 2 for TIM15)*. Added *Section 41.4.24: Using timer output as trigger for other timers (TIM16/TIM17)*. Updated *Section 41.5.8: TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	Updated *Table 293: Output control bits for complementary OCx and OCxN channels with break feature (TIM15)*. Updated *Section 41.6.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)*. Updated *Table 295: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)*. **LPTIM:** Updated *Section 43.3: LPTIM implementation*. Updated *Table 298: STM32MP153 LPTIM features*. Updated *Section 43.4.4: LPTIM reset and clocks*. Updated *Table 313: Effect of low-power modes on the LPTIM*. **STGEN:** Updated *Section 44.4.2: STGENC status register (STGENC_CNTSR)*. **USART:** Added *Section 51.6: USART in low-power modes*. Updated *Section 51.7: USART interrupts*. Updated *Table 371: USART interrupt requests*. Updated *Section 51.8.7: USART receiver timeout register (USART_RTOR)*. Updated *Section 51.8.8: USART request register (USART_RQR)*. Updated *Section 51.8.9: USART interrupt and status register (USART_ISR)*. Updated *Section 51.8.10: USART interrupt and status register [alternate] (USART_ISR)*. **SPI:** deeply revisited section. **SPDIF:** Added Note in *Section 54.2: SPDIFRX main features*. Added Note in *Table 397: Minimum spdifrx_ker_ck frequency versus audio sampling rate*. **SDMMC:** Updated *Section : Data path*. Moved *Section 56.7: Hardware flow control*. Updated *Section 56.10.2: SDMMC clock control register (SDMMC_CLKCR)*. Updated *Section 56.10.17: SDMMC IDMA buffer base address register (SDMMC_IDMABASER)*. **OTG:** Updated *Section 58.4.3: OTG core*. Updated *Section 58.4.4: Embedded full-speed OTG PHY connected to OTG*. Updated *Section 58.4.5: OTG detections*. Added *Figure 701: OTG peripheral-only connection*.

Table 599. Document revision history (continued)

Date	Revision	Changes
15-Jan-2021	5 (continued)	Added *Figure 702: OTG host-only connection*. Updated *Section 58.14.4: OTG USB configuration register (OTG_GUSBCFG)*. Updated *Section 58.14.15: OTG general core configuration register (OTG_GCCFG)*. Updated *Section 58.14.29: OTG host channel x characteristics register (OTG_HCCHARx)*. Updated *Section : Halting a channel*. **ETH:** deeply revisited section. **DEBUG:** Updated *Section 64.8: Serial wire and JTAG debug port (SWJ-DP)*. Updated *Table 567: MEM-AP registers*. Updated *Section : AP1 control/status word register (AP1_CSW)*. Updated *Section : DBGMCU identity code register (DBGMCU_IDC)*. Updated *Section : CM4ROM memory type register (CM4ROM_MEMTYPE)*. Updated *Section : CM4ROM CoreSight peripheral identity register 4 (CM4ROM_PIDR4)*. Updated *Section : CM4ROM CoreSight peripheral identity register 0 (CM4ROM_PIDR0)*. Updated *Section : CM4ROM CoreSight peripheral identity register 1 (CM4ROM_PIDR1)*. Updated *Section : CM4ROM CoreSight peripheral identity register 2 (CM4ROM_PIDR2)*. **Device electronic signature:** *Section 65.2: Device part number (RPN)*.
26-Apr-2022	6	**BSEC:** Updated *Section 3.2: BSEC main features*. Updated *Section 3.3.2: Interface to OTP*. Updated *Section 3.3.3: OTP security modes* inlcuding *Table 11: OTP modes definition*.. Updated *Section 3.3.6: OTP operations*. Updated *Section 3.6.2: BSEC OTP control register (BSEC_OTP_CONTROL)*. Updated *Section 3.6.9: BSEC OTP lock status register x (BSEC_OTP_WRLOCKEDx)*. **OTP:** Updated *Table 17: OTP list words 0 to 31*. Updated *Table 18: OTP list words 32 to 95*. DDRPHYC: Updated *Section 7.1: DDRPHYC features*. Updated *Table 22: DDRPHYC I/O list*. Updated *Section 7.5.1: PHY and SDRAM initialization*. Updated *Section 7.5.7: Read valid training (RVTRN)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
26-Apr-2022	6 (continued)	Updated *Section 7.7.2: DDRPHYC PHY initialization register (DDRPHYC_PIR)*. Updated *Section 7.7.3: DDRPHYC PHY global control register (DDRPHYC_PGCR)*. Updated *Section 7.7.4: DDRPHYC PHY global status register (DDRPHYC_PGSR)*. Updated *Section 7.7.5: DDRPHYC DDR DLL global control register (DDRPHYC_DLLGCR)*. Updated *Section 7.7.40: DDRPHYC byte lane n DLLC register (DDRPHYC_DXnDLLCR)*. **PWR:** Updated *Section 9.3.1: PWR core domain*. Updated *Table 34: Functionalities depending on system operating mode* footnote. Updated *Section 9.6.9: PWR LP-Stop and LPLV-Stop modes*. **RCC:** Updated *Figure 87: Clock distribution for RTC*. HSEM: Updated *Section 11.4: HSEM registers* intoduction. **IPCC:** Updated *Figure 112: IPCC Simplex - Send procedure state diagram*. **ETZPC:** Updated reset value in *Section 15.7.6: ETZPC version register (ETZPC_VERR)*. **MDMA:** Updated *Section 17.5.7: MDMA channel x transfer configuration register (MDMA_CxTCR)*. Updated *Section 17.5.9: MDMA channel x source address register (MDMA_CxSAR)*. Updated *Section 17.5.10: MDMA channel x destination address register (MDMA_CxDAR)*. Updated *Section 17.5.13: MDMA channel x trigger and bus selection register (MDMA_CxTBR)*. **DMA:** Updated *Section 18.5.5: DMA stream x configuration register (DMA_SxCR)*. **DMAMUX:** Updated *Section 19.4.4: DMAMUX request line multiplexer*. Updated *Section 19.4.5: DMAMUX request generator*. **CRC:** Updated *Section 24.2: CRC main features*. Updated *Figure 141: CRC calculation unit block diagram*. Updated *Section 24.3.3: CRC operation*. Updated *Section 24.4: CRC registers* introduction.

Table 599. Document revision history (continued)

Date	Revision	Changes
26-Apr-2022	6 (continued)	Added *Section 24.4.6: CRC hardware configuration register (CRC_HWCFGR)* to *Section 24.4.9: CRC size identification register (CRC_SIDR)*. **FMC:** Updated entire section **QUADSPI:** Updated *Section 26.1: Introduction*. Updated *Section 26.2: QUADSPI main features*. Updated *Figure 168: QUADSPI block diagram*. Updated *Section : Address phase*. Updated *Section : Data phase*. Updated *Section : FIFO and data management*. Updated *Section 26.3.6: QUADSPI automatic status-polling mode*. Updated *Section : Automatic status-polling mode*. **ADC:** Updated *Section : Constraints between ADC clocks*. Updated *Section : DMA one shot mode (DMNGT=01)*. Updated *Figure 216: AUTDLY=1 in auto- injected mode (JAUTO=1)*. Updated *Figure 210: Example of overrun (OVRMOD = 0)*. Updated *Figure 225: Regular and injected oversampling modes used simultaneously*. Updated *Figure 226: Triggered regular oversampling with injection*. Updated *Section : Regular simultaneous mode with independent injected*. Updated *Section : Reading the temperature*. **DTS:** Updated *Figure 248: Temperature sensor functional block diagram*. Updated *Table 205: Trigger configuration*. Updated *Table 29.6.4: Temperature sensor interrupt threshold register 1 (DTS_ITR1)*. **DAC:** Updated *Section 30.2: DAC main features*. **VREFBUF:** Updated *Section 31.2: VREFBUF functional description*. Updated *Section Table 218.: VREF buffer modes*. **DSI:** Addded *Section 36.12.2: D-PHY HS2LP and LP2HS durations*. **RNG:** Updated *Section 35.2: RNG main features*. Updated *Section 35.7.3: RNG data register (RNG_DR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
26-Apr-2022	6 (continued)	**HASH:** Updated *Section 37.4.7: HMAC operation*. Added Note in *Section 37.4.5: Message digest computing*. **CRYP:** Updated *Section 36.3: CRYP implementation*. Updated *Section 36.4.7: Preparing the CRYP AES key for decryption*. Updated *Section 36.7.8: CRYP masked interrupt status register (CRYP_MISR)*. **TIM1/TIM8:** Updated *Figure 335: Control circuit in normal mode, internal clock divided by 1*. Updated *Section 38.3.16: Using the break function*. Updated *Section 38.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)*. **TIM2/TIM3/TIM4/TIM5:** Updated *Section 39.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)*. Updated *Section 39.4.11: TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)*. **TIM12/TIM13/TIM14:** Updated *Figure 432: Control circuit in normal mode, internal clock divided by 1*. Updated *Section 40.4.7: TIM12 capture/compare mode register 1 (TIM12_CCMR1)*. Updated *Section 40.5.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14)*. **TIM15/TIM16/TIM17:** Updated *Figure 459: Control circuit in normal mode, internal clock divided by 1*. Updated *Section 41.4.13: Using the break function*. Added *Section 41.4.15: 6-step PWM generation*. Updated *Section 41.5.4: TIM15 DMA/interrupt enable register (TIM15_DIER)*. Updated *Section 41.5.8: TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)*. Updated *Table 293: Output control bits for complementary OCx and OCxN channels with break feature (TIM15)*. Updated *Section 41.5.16: TIM15 break and dead-time register (TIM15_BDTR)*. Updated *Section 41.6.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
26-Apr-2022	6 (continued)	Updated *Table 295: Output control bits for complementary OCx and OCxN channels with break feature (TIM16/17)*. Updated *Section 41.6.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. **LPTIM:** Updated *Section 43.2: LPTIM main features*. Updated *Section 43.4.4: LPTIM reset and clocks*. Updated *Section 43.4.7: Trigger multiplexer*. Updated *Section 43.4.15: Encoder mode*. Updated *Section 43.7.1: LPTIM interrupt and status register (LPTIM_ISR)*. Updated *Section 43.7.4: LPTIM configuration register (LPTIM_CFGR)*. Updated *Section 43.7.5: LPTIM control register (LPTIM_CR)*. **IWDG:** Updated *Figure 504: Independent watchdog block diagram*. Updated *Section 46.4.4: Window option*. Updated *Figure 506: Reset timing due to refresh in the not allowed area*. Updated *Section 46.5: IWDG low power modes*. Updated *Section 46.6: IWDG interrupts*. Updated *Section 46.7.1: IWDG key register (IWDG_KR)*. Updated *Section 46.7.7: IWDG hardware configuration register (IWDG_HWCFGR)*. **WWDG:** Removed WWDG hardware configuration register (WWDG_HWCFGR). **RTC:** Updated *Section 48.3.14: RTC smooth digital calibration*. Updated *Section 48.6.22: RTC status clear register (RTC_SCR)*. **TAMP:** Updated *Section 49.6.1: TAMP control register 1 (TAMP_CR1)*. **USART/UART:** Updated *Section 51.5.21: USART low-power management*. Updated *Section 51.8.3: USART control register 2 (USART_CR2)*. Updated *Section 51.8.9: USART interrupt and status register (USART_ISR)*. Updated *Section 51.8.10: USART interrupt and status register [alternate] (USART_ISR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
26-Apr-2022	6 (continued)	**SAI:** Updated *Section 53.4.10: PDM interface*. Updated *Table 388: TDM frame configuration examples*. Updated *Section 53.6.19: SAI PDM delay register (SAI_PDMDLY)*. **SPDIF:** Updated *Figure 628: SPDIFRX block diagram*. Updated *Table 395: SPDIFRX pins*. Updated *Figure 633: SPDIFRX decoder*. Added *Table 398: Conditions of spdifrx_symb_ck generation*. **MDIOS:** Updated *Section 55.3.4: MDIOS data*. Updated *Section 55.3.5: MDIOS APB frequency*. Updated *Section 55.3.6: Write/read flags and interrupts*. Updated *Section 55.3.7: MDIOS error management*. Updated *Section 55.4.1: MDIOS configuration register (MDIOS_CR)*. Updated *Section 55.4.6: MDIOS status register (MDIOS_SR)*. Updated *Section 55.4.7: MDIOS clear flag register (MDIOS_CLRFR)*. **SDMMC:** Updated *Section : Data FIFO*. Updated *Section : Stream operation and CMD12*. Updated *Section : Normal boot operation*. Updated *Section : Alternative boot operation*. **FDCAN:** Added *Table 431: CAN triggers*. Updated *Section 57.4.7: Application*. **OTG:** Updated *Table 58.14.17: OTG core LPM configuration register (OTG_GLPMCFG)*. **CEC:** Updated *Table 62.7: HDMI-CEC registers* introduction. **ETH:** Updated all *Section 61: Ethernet (ETH): gigabit media access control (GMAC) with DMA controller*. **DEBUG:** Replaced nJTRST by NJTRST.
21-Mar-2025	7	**Documentation conventions:** Added *Section 1.3: Register reset value*. **Memory and bus architecture:** Updated *Table 1: Memory map overview STM32MP153*. **BSEC:** Updated *Section 3.3.3: OTP security modes*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	**OTP:** Updated introduction of *Section 4: OTP mapping (OTP)*.Updated *Table 17: OTP list words 0 to 31*. **DDRCTRL:** Updated *Section 5.2: DDRCTRL features*. Updated *Section 5.5.3: Power saving features*. Updated introduction of *Section 5.7: DDRCTRL registers*. Updated *Section 5.7.1: DDRCTRL master register 0 (DDRCTRL_MSTR)*. Updated *Section 5.7.3: DDRCTRL mode register read/write control register 0 (DDRCTRL_MRCTRL0)*. **DDRPHYC:** Updated *Section 7.7.6: DDRPHYC AC DLL control register (DDRPHYC_ACDLLCR)*. Updated *Section 7.7.24: DDRPHYC MR3 register for DDR3 (DDRPHYC_DDR3_MR3)*. Updated *Section 7.7.40: DDRPHYC byte lane n DLLC register (DDRPHYC_DXnDLLCR)*. Added *Section 7.7.25: DDRPHYC MR3 register for LPDDR2 [alternate] (DDRPHYC_LPDDR2_MR3)*. Added *Section 7.7.26: DDRPHYC MR3 register for LPDDR3 [alternate] (DDRPHYC_LPDDR3_MR3)*. **PWR:** Updated *Section 9.3.6: PWR supply Standby sequence*. Updated *Section 9.3.8: PWR VSW domain*. Updated *Section 9.4.1: PWR power-on reset (POR)/power-down reset (PDR)*. Updated *Section 9.4.3: PWR VDDCORE OK reset*. Updated *Section 9.7: PWR WKUP pins*. Updated *Section 9.11.5: PWR MPU control register (PWR_MPUCR)*. **RCC:** Updated *Section 10.7.75: RCC DDR interface control register (RCC_DDRITFCR)*. **HSEM:** Updated *Section 11.4.8: HSEM clear semaphore key register (HSEM_KEYR)*. Updated *Section 11.4.11: HSEM version register (HSEM_VERR)*. **GPIO:** Updated *Table 82: Port bit configuration table*. Updated *Section 13.3.13: Analog configuration*. Updated *Section 13.3.15: Using the GPIO pins in the backup supply domain*. **SYSCFG:** Updated *Section 14.3.7: SYSCFG compensation cell control register (SYSCFG_CMPCR)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	**ETZPC:** Updated *Table 87: STM32MP153 TrustZone-aware peripherals*. Updated *Table 92: decprot bits for securable and Isolable Peripherals (type 3) and behavior*. Updated *Section 15.7.3: ETZPC securable peripheral control register x (ETZPC_DECPROTx)*. Updated *Section 15.7.7: ETZPC identification register (ETZPC_IDR)*. Updated *Section 15.7.8: ETZPC size identification register (ETZPC_SIDR)*. **DMA:** Updated *Section 18.3.19: Error management*. **DMAMUX:** Updated *Section 19.4.3: DMAMUX channels*. Updated *Section 19.4.4: DMAMUX request line multiplexer*. Updated *Section 19.4.5: DMAMUX request generator*. **GIC:** Updated *Section 22.5.1: GICD control register (GICD_CTLR)*. **CRC:** Updated *Section 24.4.6: CRC hardware configuration register (CRC_HWCFGR)*. **FMC:** Updated *Section 25.5: AHB interface*. Updated *Figure 158: Wait configuration waveforms*. Updated *Figure 159: Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)*. Updated *Figure 160: Synchronous multiplexed write mode waveforms - PSRAM (CRAM)*. Updated *Section 25.8.6: NOR/PSRAM controller registers*. Updated *Table 167: Programmable NAND flash access parameters*. Updated *Figure 161: NAND flash controller waveforms for common memory access*. Updated *Section 25.9.4: NAND flash operations*. Updated *Section 25.9.5: NAND flash prewait function*. Updated *Figure 165: Example of 2 Kbyte page program sequence with 4-bit BCH*. **QUADSPI:** Updated *Section : Triggering the start of a command*. **ADC:** Updated *Figure 178: ADC clock scheme*. Updated *Figure 184: Enabling / Disabling the ADC*. Updated *Section 28.4.10: Constraints when writing the ADC control bits*. Updated *Section : I/O analog switch voltage booster*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	Updated *Figure 185: Analog-to-digital conversion time*. Updated *Section 28.4.20: Injected channel management*. Updated *Section 28.4.22: Queue of context for injected conversions*. Updated *Section : Analog watchdog*. Updated *Figure 235: Alternate trigger: 4 injected channels (each ADC) in discontinuous mode*. Updated *Figure 245: VBAT channel block diagram*. Updated *Section 28.6.4: ADC configuration register (ADC_CFGR)*. **ADC:** Updated *Section 28.6.19: ADC analog watchdog 2 configuration register (ADC_AWD2CR)*. Updated *Section 28.6.20: ADC analog watchdog 3 configuration register (ADC_AWD3CR)*. Updated *Section 28.7.5: ADC hardware configuration register (ADC_HWCFGR0)*. Updated *Section 28.7.6: ADC version register (ADC_VERR)*. Updated *Section 28.7.7: ADC identification register (ADC_IPIDR)*. Updated *Section 28.7.8: ADC size identification register (ADC_SIDR)*. **DTS:** Updated *Section 29.3.7: Temperature measurement principles*. Updated *Section 29.3.9: Quick measurement mode*. Updated introduction of *Section 29.5: DTS interrupts*. **DAC:** Updated *Figure 252: Dual-channel DAC block diagram*. Updated *Figure 257: DAC conversion (SW trigger enabled) with LFSR wave generation*. Updated *Figure 259: DAC conversion (SW trigger enabled) with triangle wave generation*. Updated *Section 30.4.12: DAC channel buffer calibration*. Added *Section 30.4.13: DAC channel conversion modes*. Added *Section 30.7.1: DAC control register (DAC_CR)*. Added *Section 30.7.21: DAC hardware configuration register (DAC_HWCFGR0)*. **DFSDM:** Updated *Section 32.4.3: DFSDM reset and clocks*. Updated formula in *Section : Manchester coded data input format operation*. Updated formula in *Section : Clock absence detection*. Updated formula in *Section : Manchester/SPI code synchronization*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	**DCMI:** Updated *Section : Data formats*. **DSI:** Updated *Section 36.3: DSI Host main features*. Updated *Section 36.4.4: System level architecture*. Updated *Section 36.5: Functional description: video mode on LTDC interface*. Updated *Section : Synchronization with the LTDC*. **DSI:** Updated *Section 36.14.5: Configuring the DSI Host LTDC interface*. Updated *Section 36.15.6: DSI Host LTDC polarity configuration register (DSI_LPCR)*. Updated *Section 36.16.1: DSI Wrapper configuration register (DSI_WCFGR)*. Updated *Section 36.16.5: DSI Wrapper interrupt flag clear register (DSI_WIFCR)*. Updated *Section 36.16.8: DSI Wrapper regulator and PLL control register (DSI_WRPCR)*. **RNG:** Added *Table 250: RNG configurations*. Updated *Section 35.6.3: Data collection*. **CRYP:** Updated *Section : AES GCM processing*. Updated *Section : AES GMAC processing*. **TIM1/TIM8:** Updated *Section 38.3.18: Clearing the OCxREF signal on an external event*. Updated *Figure 363: Retriggerable one pulse mode*. Updated *Section 38.3.22: Encoder interface mode*. Updated *Section 38.3.25: Interfacing with Hall sensors*. Updated *Section 38.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)*. Updated *Section 38.4.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)*. **TIM2/TIM3/TIM4/TIM5:** Updated *Section 39.3.1: Time-base unit*. Updated *Section 39.3.12: Clearing the OCxREF signal on an external event*. Updated *Figure 408: Retriggerable one-pulse mode*. Updated *Section 39.3.15: Encoder interface mode*. Added *Section : Slave mode: Combined reset + trigger mode*. Updated *Section 39.4.8: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1) (x = 2 to 5)*. **TIM12/TIM13/TIM14:** Updated *Figure 443: Retriggerable one pulse mode*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	Updated *Section 40.4.8: TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1)*. Updated *Section 40.5.6: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13 to 14)*. **TIM15/TIM16/TIM17:** Updated *Figure 478: Retriggerable one pulse mode*. Updated *Section 41.6.7: TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)*. **STGEN:** Updated *Figure 502: STGEN block diagram*. Updated *Section 44.4.16: STGENC component ID2 register (STGENC_CIDR2)*. **Watchdogs:** Updated *Section 45.2: Watchdog brief functional description*. Updated *Figure 503: Watchdogs top view*. **IWDG:** Added *Table 319: IWDG delays versus actions*. Updated *Section 46.4.4: Window option*. Updated *Section 46.7.4: IWDG status register (IWDG_SR)*. **WWDG:** Added *Figure 509: Watchdog block diagram*. Updated *Section 47.3.4: Controlling the down-counter*. Updated *Section 47.4: WWDG interrupts*. Updated *Section 47.5.2: WWDG configuration register (WWDG_CFR)*. **RTC:** Updated *Section 48.3.5: Clock and prescalers*. Updated *Section 48.3.17: Tamper and alarm output*. Updated *Section 48.6.19: RTC status register (RTC_SR)*. Updated *Section 48.6.20: RTC non-secure masked interrupt status register (RTC_MISR)*. Updated *Section 48.6.21: RTC secure masked interrupt status register (RTC_SMISR)*. **TAMP:** Added *Table 341: Effect of low-power modes on TAMP*. Updated *Section 49.6.7: TAMP secure mode register (TAMP_SMCR)*. **I2C:** Master and slave terms in *Section 50: Inter-integrated circuit interface (I2C)* replaced with controller and target, respectively. Entire section revised. **USART/UART:** Updated *Table 363: USART features*. Added *Table 364: USART/UART input/output pins*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	Added *Table 365: USART internal input/output signals*. Updated *Figure 548: Start bit detection when oversampling by 16 or 8*. Updated *Noise error is not supported in SPI and IrDA modes.* Updated *Figure 564: Transmission using DMA*. Updated *Figure 565: Reception using DMA*. **USART/UART:** Updated *Section 51.8.14: USART prescaler register (USART_PRESC)*. Updated *Section 51.8.16: USART hardware configuration register 1 (USART_HWCFGR1)*. **SPI:** Updated *Section 52.2: SPI main features*. Updated *Section 52.5.3: CRC computation*. Updated *Figure 586: Low-power mode application example*. Updated *Section 52.8: I2S main features*. Updated *Section 52.9.2: Pin sharing with SPI function*. Updated *Table 375: Bitfields usable in PCM/I2S mode*. Updated *Section 52.9.5: Supported audio protocols*. Added a note to *Figure 587: Waveform examples* through *Figure 596: Slave PCM waveforms*. Updated *Section 52.9.6: Additional Serial Interface Flexibility*. Updated *Section 52.9.7: Startup sequence*. Updated *Section 52.9.9: Clock generator*. Updated *Section 52.9.12: Handling of underrun situation*. Updated *Section 52.9.13: Handling of overrun situation*. Removed *Section: Master I2S MSB Aligned, full-duplex*. Updated *Section : Slave I2S Philips standard, receive*. Updated *Section 52.11.1: SPI/I2S control register 1 (SPI_CR1)*. Updated *Section 52.11.6: SPI/I2S status register (SPI_SR)*. Updated *Section 52.11.14: SPI/I2S configuration register (SPI_I2SCFGR)*. **SAI:** Updated *Section 53.1: Introduction*. Updated *Section : Configuring and enabling SAI modes*. Updated *Figure 613: PDM typical connection and timing*. Updated entire *Section 53.4.10: PDM interface*. Updated *Section 53.4.13: Specific features*. Updated *Section 53.4.16: SAI DMA interface*. Updated *Section 53.6.3: SAI configuration register 2 (SAI_ACR2)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	Updated *Section 53.6.11: SAI configuration register 2 (SAI_BCR2)*. Updated *Section 53.6.5: SAI slot register (SAI_ASLOTR)*. Updated *Section 53.6.13: SAI slot register (SAI_BSLOTR)*. **SDMMC:** Updated *Table 405: SDMMC operation modes e•MMC*. **FDCAN:** Updated *Table 429: CAN subsystem I/O signals*. Updated *Figure 683: Transceiver delay measurement*. Added *Section 57.4.2: Error management*. Updated *Table 453: Extended message ID filter element*. Updated *Section 57.5.3: FDCAN data bit timing and prescaler register (FDCAN_DBTP)*. Updated *Section 57.5.4: FDCAN test register (FDCAN_TEST)*. Updated *Section 57.5.33: FDCAN Rx buffer element size configuration register (FDCAN_RXESC)*. Updated *Section 57.5.36: FDCAN Tx buffer element size configuration register (FDCAN_TXESC)*. Updated *Section 57.7.1: Clock calibration unit core release register (FDCAN_CCU_CREL)*. Updated *Section 57.7.2: Calibration configuration register (FDCAN_CCU_CCFG)*. **OTG:** Updated *Section 58.7: OTG as a USB host*. Updated *Section 58.14.31: OTG host channel x interrupt register (OTG_HCINTx)*. **USBPHYC:** Added *Section 59.4.3: USBPHYC PHY x MONITOR register (USBPHYC_MONITORx)*. **ETH:** Updated *Table 475: Ethernet peripheral pins*. Updated *Table 476: Ethernet internal input/output signals*. Updated *Figure 732: Ethernet high-level block diagram*. Updated *Figure 733: DMA transmission flow (standard mode)*. Updated *Section : DMA reception*. Updated *Figure 735: Receive DMA flow*. Updated *Section 61.4.4: Multiple channels and queues support*. Updated *Table 506: MCD clock selection*. Updated *Section 61.6.5: Reduced gigabit media independent interface (RGMII)*. Updated *Figure 759: LPI Tx clock gating (when LPITCSE = 1)*.

Table 599. Document revision history (continued)

Date	Revision	Changes
21-Mar-2025	7 (continued)	Updated *Section 61.6.5: Reduced gigabit media independent interface (RGMII)*. Updated introduction of *Section 61.9: Ethernet programming model*. **ETH:** Added *Section 61.9.15: Programming guidelines for IEEE 1588 auxiliary snapshot*. Updated *Section : Tx queue 1 send slope credit register (ETH_MTLTXQ1SSCR)*. Updated *Section : Operating mode configuration register (ETH_MACCR)*. Updated *Section : Interrupt status register (ETH_MACISR)*. Updated *Section : Interrupt enable register (ETH_MACIER)*. Updated *Section : PMT control status register (ETH_MACPCSR)*. Updated *Section : One-microsecond-tick counter register (ETH_MAC1USTCR)*. Updated *Section : HW feature 0 register (ETH_MACHWF0R)*. Updated *Section : MDIO address register (ETH_MACMDIOAR)*. Updated *Section : Timestamp status register (ETH_MACTSSR)*. Updated *Section : PPS control register [alternate] (ETH_MACPPSCR)*. Updated *Section : PPS target time seconds register (ETH_MACPPSTTSR)*. **DBG:** Updated *Section : DP resend register (DP_RESEND)*. Updated *Section : DBG OS lock access register (DBG_OSLAR)*. **Device electronic signature:** Updated *Section 65.4: Package data register (PKG)*.

Index

A

B

C

D

E

F

G

H

I

L

M

O

P

Q

R

S

T

U

V

W